DIPLOMSKA NALOGA

Andreja Balon

UNIVERZA EDVARDA KARDELJA V LJUBLJANI Fakulteta za elektrotehniko in računalništvo

61001 Ljubljana
Tržaška 25
Telefon (061) 265 161, p. p. 592
Telex 31573 yu elefak

Št.: 235 vis/90

Datum: 15.3.1990

Andreja B A L O N

L j u b l j a n a

Diplomska naloga

NASLOV: VIZUALIZACIJA

BESEDILO: Predstavljena je vizualizacija in njene razdelitve po področjih.Opisan je vpliv vizualizacije na razvoj računalniških orodij in psihološki vidik sprejemanja slik. Pojem vizualizacije je razširjen na pojem multimedije, ki je predstavljena z izdelavo aplikacije v programskem okolju HyperCard s pomočjo jezika HyperTalk.

Diplomsko nalogo izstavil:

doc.dr.Franci SOLINA

Univerza Edvarda Kardelja v Ljubljani Fakulteta za elektrotehniko in računalništvo

D e k a n :

prof.dr.Baldomir ZAJC

VIZUALIZACIJA

Andreja Balon

DIPLOMSKA NALOGA
IZ
RAČUNALNIŠTVA IN INFORMATIKE

predložena
Fakulteti za elektrotehniko in računalništvo
Univerze v Ljubljani
kot delna izpolnitev pogoja za pridobitev naslova
diplomirani inženir računalništva

Maj 1990

VIZUALIZACIJA

Andreja Balon

Mentor:
Franc Solina

Povzetek

Diplomsko delo predstavlja področje vizualizacije , ki ga razdelimo na vizualizacijo na tradicionalnih področjih in vizualizacijo v računalništvu. Ker je za vizualizacijo pomemben psihološki vidik sprejemanja slik, ga na kratko opišemo na začetku. Vizualizacijo v računalništvu razdelimo po njenih področjih; vizualizacija v znanosti, programiranje vizualizacije in vizualni jeziki. Za vsako od naštetih področij podajamo razlago ter primere uporabe oziroma pristope k njihovi uporabi. Vizualizacija je bistven element multimedije, ki poleg slike povezuje še tekst, zvok, diagrame, animacijo ter igrani in/ali animirani video. V drugem delu je predstavljena implementacija multimedijskega okolja in kot primer je podana predstavitev in uporaba objektno orientiranega jezika *HyperTalk* v programskem okolju *HyperCard*.

VISUALIZATION

Andreja Balon

Supervisor:
Franc Solina

ABSTRACT

The present thesis entails the field of visualization which is devided into visualization along traditional lines and visualization in computer science. As the psyhological aspect of image is of vital importance for visualization, it is shortly described in the begining. Visualization in computer science is devided into three main fields: scientific visualization, program visualization and visual programming. An explanation and examples of approach to applications are given for each field. Visualization is an essential element of multimedia which, in addition to image, interconnects text, sound, diagrams, animation and motion picture as well as animated video. The second part covers implementation of multimedia environment, with presentation and application of *HyperTalk* object oriented language in *HyperCard* program environment given as an example.

Kazalo

1. Poglavje

Uvodna beseda

1.1 Na kratko o temi diplomske naloge

Že od prazgodovine je človek vezan na sliko kot orodje za izražanje svojih misli in pogledov ter kot obliko sporočila. Jamski človek je narisal sliko svojega okolja na steno jame. Antična legenda nam pripoveduje, da so Arhimeda ubili rimski vojaki medtem, ko je risal figure v pesek. Ohranjene slike izgubljenih civilizacij nam povedo ogromno o načinu življenja tedanjih ljudi. Tudi v umetnosti predstavlja slika zelo učinkovito orodje sporočanja. Tako v slikarstvu umetniki na zelo realističen ali pa tudi popolnoma abstrakten način sporočajo svoja občutja in poglede na svet. Slika govori vsem, ne glede na jezik in pisavo. Literarni teksti so s predstavitvijo v gledališču ali v filmu dobili popolnoma novo dimenzijo. Znanstveniki in konstrukterji (Da Vinci) so že v preteklosti raziskovali in ustvarjali s pomočjo slik in modelov. Kemiki so v tem stoletju pričeli s preučevanjem strukture materialov s pomočjo modelov mulekul, ki so jih naredili iz plastike in lesa. Vsi ti primeri nam potrjujejo pomen predstavitve podatkov in konstrukcij v sliki. Področje oziroma disciplina, ki se ukvarja s predstavitvijo podatkov in konstrukcij, je vizualizacija. Vizualizacija podatkov in konstrukcij ni nova in tudi ni vezana na računalnik. Človek s pomočjo slike celostno dojame informacijo, medtem ko tekst in številke sporočajo informacijo linearno oziroma zaporedno. Človek lahko sprejema velike količine informacij predstavljenih v sliki v paralelnih, zaznavnih ali sledilnih vzorcih z neverjetno hitrostjo.

Pri vizualnih predstavitvah objektov, dogodkov, modelov in podobno pa ne smemo pozabiti na način sprejemanja in zaznavanja slik vsakega posameznika—individualni psihološki vidik in obči psihološki vidik sprejemanja slik ter napotkov za uporabo barv, zvoka, definiranja funkcij tipk ter delovanja menujev in dialogov.

Razvoj vizualizacije v računalništvu je zgodovinsko pogojen s tehnološkim razvojem. Na začetku razvoja računalništva je bilo možno predstaviti rezultate računalniških obdelav samo z neskončnimi listami papirja, popisanimi s številkami, medtem ko imamo danes na razpolago vizualne izhodne enote kot so grafični terminali ter risalniki. Tudi pri vnašanju slik v računalnik danes še zdaleč nismo omejeni samo

na tipkovnico, temveč imamo na voljo miške, optične svinčnike, digitalne tabele, skanerje, grafične digitalizatorje in podobno. Shranjevanje vizualnih podatkov zahteva več spomina, toda tudi lokalni pomnilniški mediji so postali zmogljivejši: CD-ROM diski, CD-audio diski, laserski videodiski, optični diski in podobno. Poleg vizualne predstavitve rezultatov računalniških obdelav ter vnosa slik si v zadnjem času želimo tudi spremljati dogajanje med procesiranjem računalniških obdelav s pomočjo slik in na ta način računalnike še bolj približati človekovem načinu dojemanja informacij. Pomembne točke pri tem so:

1. Način dela z računalnikom (uporabniški vmesnik), preko katerega sporočimo računalniku naše zahteve.

2. Način predstavitve rezultatov obdelav računalnika.

3. Način predstavitve informacij.

Vizualizacija v računalništvu se ukvarja s premostitvijo prepada med računalnikom in uporabnikom z vidika načina dela ter predstavitve rezultatov in informacij s pomočjo slike in animacije slik.

Vizualizacija je obenem bistven element multimedije. Multimedija je kombinacija slik, teksta, diagramov, animacij, zvokov ter animiranega in/ali igranega videa. To je pojem, kjer se srečajo televizija, film, audio tehnika, založništvo in računalnik.

1.2 Pregled vsebine diplomske naloge

- V drugem poglavju je predstavljena teorija vida, s pomočjo katere definiramo proces videnja, saj vemo, da pri gledanju ni vse odvisno samo od oči, temveč tudi od našega mišljenja. S pomočjo eksperimentov je bilo ugotovljeno več dejstev o vplivu barve in zvoka na človeški način sprejemanja informacije, zato se v tem poglavju podaja tudi nekaj napotkov za pisanje uporabniških vmesnikov.

- V tretjem poglavju je podan pregled vizualizacije v računalništvu z razlago in navedbami primerov za vsako področje posebej.

- Četrto poglavje je namenjeno predstavitvi multimedije, pojmu, ki poleg slike združuje še druga medijska orodja, kot sta na primer zvok in film v nelinearno povezanem vzorcu informacij.

- Peto, zadnje poglavje pa predstavlja programsko okolje *HyperCard* s programskim jezikom *HyperTalk,* ki je primerno za izdelavo multimedijskih aplikacij, saj v sebi združuje vsa potrebna orodja.

2.

Psihološki vidik zaznavanja slik

To poglavje predstavlja proces videnja—kako vidimo in kakšno vlogo igra pri tem naše mišljenje. Pri izdelavi uporabniških vmesnikov se uporabljajo različna orodja kot so barva, zvok, funkcijske tipke in menuji, zato je v tem poglavju navedenih tudi nekaj navodil za pisanje vmesnikov.

Vizualno zaznavanje je človekovo okno v svet—njegova naloga je razpoznavanje in sporočanje objektov v okolici ter njihovih odvisnosti. Je povezava med stvarnim okoljem in našim pojmovanjem okolja in izpostavi pomembnost našega mišljenja ter zagotavlja korespondenco z zunanjim svetom. Videnje je proces sklepanja. Kaj vidimo, ko se ozremo okoli sebe, ni odvisno samo od tega, kaj lahko vidimo, temveč tudi od tega, kako je naš vizualni sistem organiziran in kako interpretira podobe, ki padejo na očesno mrežnico. Vizualni sistem je več kot samo pasivni prenosnik signalov v možgane, saj aktivno sodeluje pri njihovem organiziranju in interpretaciji. Vizualni sistem mora organizirati in interpretirati podobe na mrežnici, saj je podoba na mrežnici konsistentna z velikim številom možnih situacij v realnem svetu. Podobe na mrežnici so v bistvu dvoumne, razlog za dvoumnost je v tem, da ima svet tri dimenzije, slika na mrežnici pa le dve. Da bi svet lahko dojemali v treh dimenzijah, je potrebno vplesti zapleteno sklepanje, ki poteka večinoma povsem podzavestno [19]. Navadno vidimo in razpoznamo stvari avtomatično, brez posebnih naporov. Pravilnosti delovanja našega procesa videnja ne preverjamo. Težko je ugotoviti, kako naš vid deluje. Psihologija zaznavanja in kognitivna znanost znanstveno še nimata popolne razlage, saj je to povezano s problemom človekovega mišljenja. Da bi lahko rešili problem vida, ne smemo zanemariti nauka teorije izračunljivosti: potrebujemo najmanj tri različne nivoje razlage. Potrebujemo teorijo o tem, "kaj" se izračunava—kakšni so vhodni podatki v proces, kaj lahko dobimo iz procesa ter kakšne omejitve lahko vodijo proces. Potrebujemo teorijo o tem, "kako" sistem izvrši izračune—algoritem, ki ga uporablja. Potrebujemo tudi teorijo o implementaciji, osnovano na nevropsihologiji—zgradbo celic nevronov, v katerih je vgrajen algoritem. David Marr je združil vse tri teorije v skupno teorijo vida [21].

Empiriki razmišljajo o možganih kot o nepopisanem listu papirja, ki dobiva informacije, ki omogočajo videnje iz zunanjega realnega sveta. Racionalisti pa trdijo, da imajo možgani že vsebovano znanje o svetu. Oboji imajo prav. Informacija je vsebovana v vzorcu svetlobe, ki pada na oko, vendar je proces videnja odvisen tudi od domnev o naravi sveta, v katerem živimo. Domneve o svetu, v katerem živimo, pa lahko dobimo na dva načina. Lahko so vgrajene v živčni sistem kot rezultat razvoja človeka v preteklih miljonih let ali pa naučene v človekovem življenju. Obe vrsti domnev o svetu sta pomembni pri percepciji, vendar na različen način. Osnovne stopnje videnja, ki pripeljejo k konstrukciji videne podobe, so:

- Prva stopnja videnja je fizična interakcija med svetlobo, ki pade na mrežnico in vizualnim pigmentom v mrežničnih celicah.
- Druga stopnja videnja je lociranje sprememb v intenzivnosti.
- Tretja stopnja videnja je osnovno skiciranje podobe.

Osnovne stopnje videnja in zaznavanje globine (stereopsis) so prirojene lastnosti vida. Priučena pravila videnja pa so trodimenzionalne oblike objektov, identificiranje objektov na osnovi njihove oblike in sprejemanje njihove relativne lokacije v prostoru. Razlika med obema zvrstema znanja o svetu lahko označi mejo med čisto percepcijo in kognitivnostjo, vendar je meja lahko zelo zabrisana. Dejstvo je, da kadar sprejemamo svet, ne moremo dobiti informacije o relativni globini površin brez uporabe prirojenih lastnosti vida. Prav tako pa ne moremo identificirati objektov brez uporabe priučenega znanja o svetu [21].

2.1 Navodila za pisanje vmesnikov

Vse ugotovitve, ki so rezultat raziskav teorije vida, moramo upoštevati pri graditvi grafičnih uporabniških vmesnikov in s tem tudi v multimediji, kjer moramo upoštevati tudi vidik sprejemanja zvoka in principov učenja.

Pri pisanju vmesnikov moramo paziti na uporabo naslednjih orodij:

- barva
- zvok
- funkcije tipk na tipkovnici
- menuji

2.1.1 Barva

Kadarkoli uporabljamo barvo, se moramo zavedati, da barve ne uporabljamo zato, da je predstavljena slika videti lepše, temveč zato, da sliki dodamo nov pomen.

Barva je lahko dodatna pot za informiranje uporabnika, vendar jo moramo uporabljati previdno, saj lahko v nasprotnem primeru zavaja in prikriva informacijo. Barve naj bodo izključno domena uporabnika, kateremu naj bo omogočeno spreminjati ali odstraniti barve, ki so določene z aplikacijo. Barva naj bo podatek in ne vmesnik. Teorije o pravilni uporabi barv v aplikacijah so nepopolne oziroma nezadostno definirane. Način sprejemanja barv s pomočjo človeškega očesa in barvni subjektivni efekti niso popolnoma raziskani. V tradicionalnem oblikovanju uporabniških vmesnikov se barva uporablja za razlikovanje in združevanje objektov in informacij na naslednji način:

- razmeji različne površine,
- prikazuje funkcionalno povezanost stvari,
- prikazuje medsebojno povezavo stvari,
- označuje kritične poteze.

Različne barve imajo standardne pomene v različnih kulturah. Pomen barv je priučen in ni odvisen od valovne dolžine barve. Nekaj barv s standardnim pomenom je:

- rdeča: stop, napaka, neuspeh,
- rumena: opozorilo, previdnost ali zakasnitev,
- zelena: naprej, pripravljenost, vklop,
- tople barve so rdeče, oranžne in rumene, medtem ko so modra in zelena hladne barve.

Barve dobijo dodaten standarden pomen znotraj posameznih disciplin: v finančnem svetu pomeni rdeča barva izgubo, črna pa dobiček. Za kartografe pomeni zelena gozdnate površine, modra pomeni vodo in rumena puščave. V aplikacijah za specifična področja lahko uporabimo prednosti pomena barv, vendar naj velja načelo, da lahko uporabnik spreminja ali pa odklopi barve, določene z aplikacijo. Za vzbuditev pozornosti uporabnika sta oranžna in rdeča primernejši od ostalih barv, vendar hkrati pomenita opozorilo ali nevarnost. Čeprav je dandanes lahko na ekranu že zelo veliko število različnih barv (256 pri večini osebnih računalnikih), lahko človeško oko razlikuje samo 128 čistih barvnih odtenkov. Toda študije so pokazale, da takrat, ko uporabljamo barve za označitev informacij, lahko učinkovito spremljamo samo štiri do sedem barv na ekranu naenkrat.

2.1.2 Splošni principi uporabe barv

Pri oblikovanju aplikacij v barvah se moramo držati dveh načel [3]:

1. Aplikacijo moramo razviti v črno beli tehniki.

   Najprej moramo razviti aplikacijo v črno beli tehniki. Barva naj bo dopolnilo, ki služi za podajanje posebnih informacij za tiste uporabnike, ki imajo barvne ekrane. Barva naj ne bo edino ločilo objektov. Za ločitev objektov moramo vedno uporabljati ostala orodja kot so oblika, razporeditev objekta, vzorec ali zvok, zaradi naslednjih vzrokov:

   - ekrani: večina uporabnikov nima barvnih ekranov
   - izpisi: trenutno so izpisi v barvah še vedno nenatančni, tudi v primeru zelo kvalitetnih tiskalnikov
   - barvna slepota: velik del populacije je do določene stopnje barvno slepih (v Evropi in Ameriki okoli 8 % moških in 0.5 % žensk)
   - osvetlitev: pri slabi osvetlitvi se barve težje razlikujejo, zato morajo biti razlike med barvami večje in število barv manjše, saj ne vemo, pri kakšnih pogojih se bo uporabljala naša aplikacija.

2. Omejitev uporabe barv.

   Pri standardnih vmesnikih, ki so del naše aplikacije (menuji, okviri oken) bi morali zelo malo uporabljati barve ali pa jih popolnoma opustiti. Uporabnikovo pozornost želimo usmeriti na aplikacijo in ne na menuje. Razpoložljivost barv v vsebinskem področju aplikacije je odvisna od vrste aplikacije:

   - V grafičnih aplikacijah, ki se ukvarjajo s slikami, bi morali imeti na voljo spreminjanje in uporabljanje vseh barv.
   - Ostale aplikacije, ki imajo opravka z organizacijo informacij, pa bi morale imeti barve samo za nazornejšo predstavitev informacije. Najboljša rešitev je podati uporabniku na voljo štiri do sedem različnih barv, ki jih lahko spreminja in jim dodaja nove.

Barva dodaja novo dimenzijo možnim kontrastom, zato moramo biti posebej pozorni na dobro čitljivost in razpoznavnost. Barve v aplikaciji bodo bolj opazne, če bo ozadje sive, črne ali bele barve. Študije so pokazale, da je branje barvnega teksta dosti težje kot pa branje črnega teksta na beli podlagi. Najbolj nečitljiva barva je svetlo modra, zato se naj ne bi uporabljala za tekst, tenke črte in majhne oblike. Barve, ki se razlikujejo samo v količini vsebovane modre barve, pa naj ne bodo sosednje. Vendar pa je za stvari, za katere želimo, da so neopazne, modra barva najprimernejša (papir z vrisano milimetersko mrežo). Ljudje težko ločimo med majhnimi področji barv, zato naj bodo barvne ploskve večje. Pri spremembi barve majhnih objektov pa naj bo sprememba barve očitna. Kadar barvamo objekte, ki so že določeni v črno beli tehniki, pobarvamo samo črne objekte.

2.1.3 Zvok

Splošni načeli uporabe zvoka sta:

1. Zvok je lahko vgrajen v standardni vmesnik zato, da bi opozarjal uporabnika na stanje aplikacije.
2. Zvok je lahko uporabljen za nadzorovanje uporabnika pri nepričakovanih dogodkih oziroma takrat, kadar uporabnik ne gleda v ekran.

Kadar obdeluje računalnik časovno obsežno obdelavo in je lahko uporabnik odvrnil pogled z ekrana, je zvok zelo primeren pripomoček za obvestilo uporabniku o zaključku obdelave oziroma o tem, da proces potrebuje sodelovanje uporabnika. Kadar ima aplikacija različna stanja ali načine dela, ima lahko vsak vstop ali zaključek dela na posameznem nivoju svoj zvok. To lahko poudari trenutno stopnjo aplikacije in prepreči zmedo.

Splošna navodila uporabe zvoka

- *Ne pretiravajmo*

  Premislimo kje in kako uporabljati zvok v naši aplikaciji. Če pretiravamo pri uporabi zvoka, zvok ne bo dodal novega pomena vmesnika, ampak bo verjetno nadlegoval uporabnika.

- *Zvok naj bo povezan z dogodkom*

  Zvok naj ne bo nikoli edini indikator dogodka; vedno naj bo tudi vizualna indikacija dogodka, še posebej, če mora uporabnik vedeti kaj se dogaja. Razlog je v tem, da lahko uporabnik zvok izključi, ali pa je zvok izven slišnega področja računalnika oziroma ima zvočnik v okvari.

- *Naravnost in nevsiljivost zvoka*

  Večina zvokov je lahko subtilnih, pa vendar dosežejo svoj namen. Glasni, parajoči zvoki so lahko neprijetni ali zastrašujoči. Vedno moramo preverjati zvok pri sebi in pri drugih ljudeh kar nekaj časa (dva tedna), preden ga vključimo v aplikacijo. Prav tako bi se morali izogibati melodij ali preprostih pesmic, saj lahko več kot trije toni melodije postanejo nadležni oziroma zvenijo neumno, če jih pogosto poslušamo.

- *Opazne razlike med zvoki*

  Uporabniki se lahko naučijo razlikovati in ločiti med zvoki, vendar naj bo razlika med različnimi zvoki očitna. Nemuzikalni ljudje pogosto ne ločijo med dvema podobnima notama ali akordoma, posebno ne, če je med njima dolg časovni razmik.

- *Uporabniški nadzor zvoka*

  Uporabnik naj ima vedno možnost za izključitev zvoka oziroma, če računalnik dopušča, spremembo jakosti zvoka.

- *Možnost spreminjanja zvoka*

  Vedno shranimo zvok kot podatek in omogočimo uporabniku, da spremeni zvok oziroma melodijo.

2.1.4 Funkcije tipk na tipkovnici

Funkcije tipk na tipkovnici so nekako že standardno definirane glede na določene programske jezike in urejevalnike tekstov. Pri določanju novih funkcijskih tipk se moramo zavedati, da lahko uporabnika zmedemo in mu otežimo delo, če določimo posebne funkcije za standardne tipke (Esc ponavadi pomeni razveljavitev odločitve).

2.1.5 Menuji

Ločimo dve vrsti menujev:

1. *Hierarhični menuji*

   Hierarhični menuji se logično širijo tako, da je lahko element menuja hkrati naslov podmenuja. Za zagotovitev enostavnosti in jasnosti menujev moramo upoštevati naslednje:

   - Hierarhični menuji naj bodo uporabljeni samo za listo elementov, ki so med sabo funkcijsko povezani (stil črk in njihova velikost)
   - Uporabljali naj bi samo enonivojske hierarhične menuje, čeprav imamo na voljo večje število nivojev. Ta dodaten nivo menuja potencialno poveča število elementov, ki jih lahko uporabnik izbere iz menuja. Če bi potrebovali več nivojev, je aplikacija verjetno zelo kompleksna in potrebno je še enkrat premisliti naš koncept oblikovanja aplikacije.

2. *"Pop-up" menuji*

   Ti menuji se pojavijo kot posebno okno, v katerega določamo oziroma izbiramo določene podatke. Ti menuji delujejo prav tako kot ostali; uporabnik lahko izbere katerikoli element, ki je na voljo in v primeru, da je menu prevelik, da bi se v celoti prikazal na ekranu se vsebina v menuju premakne navzgor oziroma navzdol, kakor želi uporabnik. Pri uporabi "pop-up" menujev moramo upoštevati naslednje:

   - Uporabljajo naj se za listo vrednosti ali povezanih elementov in nikoli za ukaze.

- Meje menuja naj bodo opazne in naj jasno pokažejo uporabniku, da vpisuje v "pop-up" menu.
- Naslov "pop-up" menuja naj bo jasno označen (invertiran), da je v primeru uporabe večih "pop-up" menujev jasno razvidno katerega uporabljamo [2].

V naslednjem poglavju je predstavljena vizualizacija v računalništvu, ki upošteva vse zgoraj naštete rezultate raziskav in eksperimentov, na vseh področjih vizualizacije v računalništvu: vizualizaciji v znanosti, programiranju vizualizacije in vizualnem programiranju.

3. Poglavje

Vizualizacija

To poglavje predstavlja pojem vizualizacije v splošnem pomenu in bolj podrobno pojem vizualizacije v računalništvu, ki je razdeljen na naslednja podpodročja:

- **Vizualizacija v znanosti**
- **Programiranje vizualizacije**
- **Vizualno programiranje**

Vizualizacija je grafična predstavitev objektov, procesov, informacij, podatkov, vsega, kar se da pretvoriti iz teksta, številk in meritev v sliko z namenom boljšega hitrejšega in celostnega razumevanja. Sestavljene barvne slike z različno intenzivnostjo, prozornostjo, obdelane z velikim številom drugih tehnik lahko, če so primerno pripravljene in obrazložene, sporočijo ogromne količine informacij v zelo kratkem času. Medtem, ko slike prinašajo visok nivo komunikacije, pa več slik v primernem dinamičnem zaporedju, lahko prispevajo še več kot samo seštevek slik. Animacija je pomembna dodatna dimenzija v vizualizaciji. Človek lahko razbere določeno informacijo skozi animacijo, ki bi jo, če bi opazoval vsako sliko posebej, ne odkril.

V psihologiji je znan naslednji eksperiment. V temni sobi so naredili nekaj posnetkov osebe, kateri so na glavne sklepe (rame, komolec, zapestje, bok, koleno in gleženj) namestili majhne lučke. Iz dinamičnega zaporedja slik se zlahka ugotovi, da slike predstavljajo človeka v gibanju, medtem, ko vsaka slika za sebe ne sporoča ničesar [19].

Potencialne prednosti grafike in sodelovanja med človekom in računalnikom sta Steve Coons in njegov študent Ivan Shuterland spoznala že pred več kot četrt stoletja. Leta 1964 je Coons v nekem intervjuju dejal:

"Videli boste raziskovalca, ki učinkovito rešuje problem, korak za korakom, in na začetku ne bo natančno vedel, kaj je njegov problem in niti ne bo natančno vedel,

kako ga bo rešil. Toda postopoma bo raziskoval ideje in skupaj z računalnikom, v popolnem sodelovanju, bosta problem rešila."

Z združevanjem področij, nekaterih zelo starih (geometrija) in nekaj relativno novih (računalniška grafika), raziskovalci razvijajo nova orodja, ki nam bodo dovolila še bolj popolno sodelovanje z računalnikom. Tako bomo morda bolje razumeli svet v katerem živimo.

Slika nam kaže kratko zgodovino vizualizacije, ki je bila zgodovinsko vezana na razvoj grafične strojne opreme.

Slika 3.1: Zgodovina vizualizacije je zgodovinsko vezana na razvoj grafične strojne opreme.

Področje vizualizacije lahko delimo na različne načine. Kljub temu vedno prihaja do prepletanja posameznih podpodročij. Ena od možnih delitev je:

1. Vizualizacija na tradicionalnih področjih, kjer računalnik omogoča hitrejše, boljše delo, saj predstavlja orodje za delo (urejevalniki tekstov, CAD/CAM sistemi, umetnost in podobno)

2. Vizualizacija v računalništvu

   - Vizualizacija v znanosti *(Scientific visualization - ViSC)*

     To področje se nanaša na animacijo podatkov, ki so rezultat superračunalniških simulacij, satelitov in merilnih naprav, uporabljenih v astronomiji, meteorologiji, geologiji in medicini.

   - Programiranje vizualizacije *(Program visualization)*

     To področje se drugače imenuje tudi animacija algoritmov in uporablja slike za predstavitev določenih vidikov izvajanj programov.

- Vizualno programiranje *(Visual programming)* V to področje spadajo jeziki, s pomočjo katerih je mogoče programirati v notaciji, ki uporablja dve ali tri dimenzije, kot so diagrami potekov (flowcharts), grafi, diagrami ali ikone.

V naslednjih treh podpoglavjih si bomo bolj natančno ogledali področja vizualizacije v računalništvu.

3.1 Vizualizacija v znanosti

V tem poglavju je predstavljena vizualizacija v znanosti, ki je postala pomemben pripomoček znanstvenikom pri raziskovanju, razumevanju in učenju. Vizualizacija v znanosti se ukvarja predvsem z vizualno predstavitvijo rezultatov računalniških simulacij, meritev in podatkov, ki jih pošiljajo sateliti in druge podobne naprave.

Vizualizacija v znanosti ni več samo zelo uporaben dodatek pri znanstvenem in tehničnem napredku, temveč je zelo pomembna v mnogih disciplinah. Kompleksni računski modeli, ki se izvajajo na superračunalnikih proizvajajo simulirane podatke v količinah, ki zahtevajo geometrično osnovane algoritme za interpretacijo. Slika sicer zahteva veliko spomina, toda visoka koncentracija podatkov upraviči uporabo vizualizacije. Vizualizacija je oblika komunikacije, ki premosti aplikacijske in tehnološke meje.

Vizualizacija v znanosti je

- *orodje za raziskovanje in razumevanje*

  V poplavi podatkov, ki jih generirajo superračunalniki in ostali zelo obsežni izvori podatkov (npr. sateliti) je nemogoče, da uporabnik količinsko preveri več kot le majhen delček podanih rezultatov—kar pomeni, da je nemogoče raziskovati v številkah predstavljene rezultate. S pojavom grafike lahko raziskovalci prevedejo celotna področja spremenljivk (predstavitev gostote, pritiska, hitrosti, entropije in podobno) v barvne slike. Kot primer naj vzamemo opazovalca, ki lahko vidi vzorce tekočin v vizualizaciji hidrodinamičnih izračunov, medtem, ko so ti vzorci nevidni, če pregledujemo več sto tisoč številk, od katerih vsaka predstavlja vrednost polja v določenem trenutku.

- *orodje za komunikacijo in učenje*

  Mnogi moderni raziskovalci niso več zadovoljno z izpiski tiskalnikov. DNA sekvence, molekularni modeli, medicinske slike, slike možganov, simulacija poletov, simulacija pretoka tekočin—vsi ti primeri morajo biti čez čas izraženi in predstavljeni vizualno. Za razumevanje, raziskovanje ali povezovanje pojavov želijo znanstveniki opazovati pojave, odvisne od časa, narediti serijo

slik, ki ilustrirajo notranje povezave različnih parametrov v določeni časovni periodi, preslikati te slike na lokalno delovno postajo za analiziranje in posneti ter ponovno predvajati eno ali več sekund animacije.

Obstajajo tri stopnje zahtevnosti vizualizacije podatkov za numerične simulacije:

1. Naknadno procesiranje *(postprocesing)*. Nanaša se na opisovanje podatkov potem, ko so bile numerične simulacije že končane.

2. Sledenje *(tracking)*. Zagotavlja prikaz rezultatov v realnem času s tem, da omogoča zaključitev v primeru nepravilne simulacije

3. Krmarjenje *(steering)*. Dovoli "dogovarjanje" z računalniškim postopkom tako, da se lahko spreminjajo vrednosti parametrov simulacije med samim izvajanjem postopkov.

Do danes je bilo največ truda vloženega v tehnike za efektivno naknadno procesiranje: metode za prenašanje simulacij podatkov k uporabnikom in nato prikaz podatkov ter razpolaganje z njimi. Sledenje je operacija, ki zahteva zmogljive paralelne računalnike za predstavitev rezultatov v realnem času in je zato zelo draga v primerjavi z rezultati, ki jih daje. Krmarjenje ponuja velike potencialne možnosti, vendar morajo biti pred tem rešeni problemi strojne opreme in uporabniških orodij. Krmarjenje sicer ni nova metoda, vendar kompleksnost današnjih simulacij zviša njegovo vrednost. Medtem ko je trenutno poudarek na superračunalniškem krmarjenju, so morda "grafični minisuperračunalniki" z njihovo računsko in grafično močjo pravo področje za učenje efektivnega krmarjenja kompleksnih simulacij.

3.1.1 Primeri uporabe vizualizacije v znanosti

- *Modeliranje mulekul*

  Uporaba računalniške grafike za vpogled v kemijsko kompleksnost se je začela že leta 1964. Interaktivna grafika je danes sestavni del akademskih in industrijskih raziskav molekularnih struktur. Metodologija je bila uspešno kombinirana s superračunalniki za modeliranje kompleksnih sistemov. Trenutno se lahko generirata dva tipa slik:

  - resnične slike molekul,
  - 3D risbe.

  Lep primer je umetniška interpretacija virusa HPV skupine umetnikov v Chicagu, ki so ocenili matematično osnovo in kompleksnost notranjega delovanja med atomi [9] ter 3D risba rhinovirusa, ki prikazuje geometrično strukturo in kompleksnost virusa.

- *Slike v medicini*

  Znanstveno računalništvo, uporabljeno za obdelavo medicinskih slik, je ustvarilo dobre možnosti v diagnostični medicini, planiranju kirurških posegov za ortopedske proteze in planiranju zdravljenja z radiacijo. V vseh primerih se možnosti kažejo v 2D ali 3D vizualizaciji delov telesa, ki so bili pred tem nedostopni za pogled.

  Primer je skanirana magnetno resonančna prispodoba (MRI) človeške glave, ki predstavlja mešanico površinske in plastno-presečno osnovane tehnike. Zunanjost glave je predstavljena z osenčenjem površine, notranjost pa s plastno-presečno tehniko. Tak pristop se uporablja za določanje pozicije notranjih struktur, kot je tumor, glede na zunanje orientacijske točke.

- *Slike in funkcije možganov*

  Za predstavitev človeških možganov se danes že uporabljajo vizualizacijske metode in računalniške slike, ki omogočajo natančno lokacijo poškodb možganov glede na orientacijske točke.

  Primer je slika obrisa možganov glede na orientacijske točke, kot so interhemisferične reže ter poškodbe možganov. Metoda določanja takih slik je bila testirana na več kot 1200 slikah devetnajstih pacientov.

- *Matematika*

  Vizualizacija pomaga matematikom razumeti kompleksne enačbe in sisteme enačb.

  Primer je programsko okolje *Mathematica* za analitično in numerično reševanje matematičnih izrazov.

- *Meteorologija*

  Študij silovitih neviht s pomočjo opazovanja in modeliranja pomaga raziskovalcem v meteorologiji razumeti pogoje v atmosferi, ki povzročijo velike rušilne tornade, ter mehanizem, s pomočjo katerega se tornadi oblikujejo in ohranjajo [37]. Študij povezav različnih okolij, označenih s spremenljivkami vertikalnega vetra, temperaturo, pritiskom in vlažnostjo zagotavlja uporabno smer nadaljnega raziskovanja.

  Primer je animirana simulacija nevihte nad Kansasom (ZDA), v kateri je površina deževnih oblakov predstavljena z mnogokotniki. Simulacija razkrije osnovne spremembe v strukturi deževnih oblakov.

- *Raziskovanje vesolja*

  Študij planetov zahteva tudi ogromno podatkov o planetih v sončnem sistemu. Znanstvenikom je sedaj na voljo zadosti podatkov, da lahko opazujejo pojave in teorijo tudi z vidika drugih področij kot so meteorologija, geografija, planetarna fizika, astronomija in astrofizika.

Primera sta animirana simulacija dinamike magnetosfere planeta Uran (simulacija prikazuje, da je kot dipolne osi odmik od planetovega kota rotacije) in simulacija srečanja *Voyager*-ja 2 in planeta Neptun, ki se je zgodilo ob koncu poletja 1989 (simulacija prikazuje pot *Voyager*-ja 2 gledano iz zemlje).

- *Astrofizika*

  Računalniški astrofiziki pri NCSA delajo skupaj s slikarji in poskušajo videti nevideno in ustvariti vizualne vzorce za pojave, ki nimajo poznane vizualne predstavitve.

  Primer je slika diagrama Schwarzschildove črne luknje in njeno obnašanje v gravitacijskem polju, ki je bila narejena s pomočjo numerične rešitve Einsteinovih numeričnih relativnostnih enačb.

- *Dinamika tekočin*

  Izračuni astronomov se opirajo na superračunalniške in vizualizacijske tehnike, da bi razumeli, zakaj se curki nekaterih galaksij razširijo na svojevrsten način. Za reševanje enačb uporabljajo magnetohidrodinamično kodo, ki opisuje valovanje tekočin ali plina[38,40].

  Primer je slika vizualizacije potovanja kozmičnega curka skozi sunek vala.

Vizualizacija pa se seveda uporablja tudi na drugih področjih v računalništvu. Tako je v naslednjem podpoglavju predstavljeno programiranje vizualizacije, področje v računalniški vizualizaciji, ki se je razvilo zaradi želje uporabnikov po spremljanju in razumevanju procesov in algoritmov, ki se izvajajo v računalniku.

3.2 Programiranje vizualizacije

Programiranje vizualizacije je področje v računalniški vizualizaciji, ki se ukvarja predvsem z animacijo algoritmov in predstavitvijo procesov, ki se odvijajo v računalniku.

Na tem področju raziskujemo vizualizacijo v smislu razumevanja programov; enostavnih in takih, ki so sestavljeni iz velikega števila sočasnih procesov. Veliko število podatkov, ki jih dobimo med izvajanjem predvsem sočasnih procesov, preseže človeške sposobnosti za sprejemanje podatkov v tekstovni obliki. Namesto da obravnavamo vizualizacijo kot mehanizem, ki ga pokličemo vedno, kadar se zgodijo spremembe in želimo te spremembe predstaviti, lahko vizualizacijo definiramo kot abstrakten pojem—preslikavo iz računskega stanja v stanje grafičnega objekta, ki ga predstavimo s pomočjo grafične naprave. Če obravnavamo vizualizacijo kot mehanizem, ki ga pokličemo za predstavitev rezultatov, potem vstavimo klice procedur na primerna mesta v kodo povsod, kjer se sproži zahteva za predstavitev rezultatov. Če pa vzamemo vizualizacijo kot preslikavo, pa ni potrebno spreminjati izvorne kode in podatke predstavimo na abstraktnem nivoju. Tak pristop k vizualizaciji je lahko uporabljen pri poljubnem programu, napisanem v poljubnem jeziku.

Raziskovanje v smeri programiranja vizualizacije je bilo motivirano z željo, da se, s pomočjo animacije, razloži delovanje algoritmov. Prvi sistem, ki je bil narejen s tem namenom, je sistem *Balsa* [6]. *Balsa* je eden prvih vizualizacijskih sistemov in je verjetno še vedno najbolj poznan in najbolj vpliven. *Balsa* uporablja koordinatorja za konstruiranje vizualizacije. Koordinator se odloči kateri dogodki v izvajanju programa se morajo zaznati in kako bodo predstavljeni. Koordinator nato razširi algoritem s klici procedur, shranjenih v knjižnici, ki so povezane z mehanizmom za generiranje slik. Ti klici procedur, ki so vsebovani v obstoječi kodi algoritma, v točkah, kjer se zgodijo ključni dogodki, preklopijo na predstavitev rezultatov.

Tak pristop, čeprav dokaj uspešen, ima slabe lastnosti zato, ker mora koordinator spemeniti programsko kodo. Prav tako je težko spremeniti podatke, ki so predstavljeni in tudi način predstavitve podatkov. Čeprav mnogo sistemov dovoli uporabniku, da izbira med večimi abstrakcijami, ki jih opravlja koordinator, ni možno, da uporabnik določiti popolnoma novo abstrakcijo, kajti to bi zahtevalo nove dogodke, katere bi bilo potrebno označiti v kodi. Kot odgovor na te težave se kaže najnovejša tendenca uporabe vizualizacije kot preslikave v animaciji algoritmov. V mnogih sistemih koordinator algoritma določa število grafičnih objektov—najbolj pogosto so to ikone—s parametri, ki se lahko spreminjajo s programskimi operacijami. *Aladdin system* [18] uporablja tak pristop. Vendar mora tudi tu *Aladdin*-ov koordinator spremeniti programsko kodo, če želi spremembo objektov. Ostalim sistemom je s povezovanjem parametrov objektov s programskim spremenljivkami uspelo odstraniti koordinatorjevo potrebo po spremembi programske kode. Spremembe spremenljivk se prenesejo na vizualizacijski sistem, ki spremeni ikone in

popravi sliko na ekranu. *Provide system* [25] je sistem, ki je namenjen odkrivanju napak in upošteva vse potencialno zanimive vidike obnašanja algoritmov. Zato klici procedur, ki so vstavljeni s pomočjo prevajalnika, avtomatično posnamejo vse spremembe stanj. *PVS system* [11], ki je namenjen za spremljanje proizvajalnega procesa domneva, da so vse zanimive informacije shranjene v bazi podatkov in zato dostopne vizualizacijskemu sistemu.

3.2.1 Pristop k programiranju vizualizacije

Vizualizacija naj bo funkcija, ki iz nekega *računskega stanja* vodi k sliki. To funkcijo bomo imenovali kot vizualizacijsko funkcijo. Glavni del našega vizualizacijskega modela [29] je predstavitev domene in definicijskega območja funkcije ter naloge funkcije. Vizualizacijska funkcija preslika množico vseh stanj v množico vseh slik.

$\mathcal{V}$: Stanja $\longmapsto$ Slike

Funkcijo $\mathcal{V}$ lahko določimo, vendar pa nas, ker želimo uporabljati vizualizacijo predvsem kot orodje za razumevanje izračunov s pomočjo pravilne abstrakcije, zanima narava te abstrakcije—pretvorba stanja informacije v simbolično obliko. Problem predstavitve simbolov s slikami je manj pomemben, čeprav vsebuje nekaj zanimivih in težkih problemov. Zato razdelimo vizualizacijsko funkcijo na dva dela ; abstrakcijsko funkcijo $\mathcal{A}$ in interpretacijsko funkcijo $\mathcal{I}$. Formalni opis je naslednji:

$\mathcal{A}$: Stanja $\longmapsto$ Objekti

$\mathcal{I}$: Objekti $\longmapsto$ Slike

Vizualizacijska funkcija $\mathcal{V}$ je združitev teh dveh funkcij.

$\mathcal{V} = \mathcal{A} * \mathcal{I}$

Za specifikacijo abstrakcijske funkcije potrebujemo nekaj informacij o njeni domeni (stanja) in definicijskem območju (objekti). Stanja so množica vseh možnih računskih stanj. V mnogo primerih je stanja težko določiti in še težje ujeti, saj so odvisna od mnogih podatkov in dogodkov. To vnaša dodatno nepotrebno kompleksnost procesu vizualizacije. Definicijsko območje abstrakcijske funkcije (objekti) je množica vseh možnih simboličnih predstavitev stanj—vsaka taka predstavitev je množica primitivnih grafičnih objektov. Razširimo predstavitev objekta s predstavitvijo vsakega grafičnega objekta z uporabo naslednje notacije: tip(parameter). Tip določa splošni razred objektov (krog, črta, pravokotnik) in parameter specificira določen objekt s podajanjem pozicije, velikosti, orientacije, barve in podobno. Sedaj določamo abstrakcijsko funkcijo kot množico vizualizacijskih pravil v naslednji obliki:

spremenljivka: vprašanje v prostoru podatkov $\Rightarrow$ lista odvisnih objektov

Spremenljivkam je vrednost določena implicitno. Za vsako stanje je množica odvisnih grafičnih objektov, ki jih dobimo, združitev vseh množic, ki jih dobimo z vsakim vizualizacijskim pravilom.

Naloge programiranja vizualizacije so naslednje:

1. Razložiti nam mora obnašanje algoritmov.
2. Predstaviti nam mora varnost algoritma, ki jo določajo pogoji v katerih mora program delovati pravilno.
3. Predstaviti mora zahtevnost algoritma oziroma kaj mora algoritem narediti.

Vizualizacija v znanstvenem raziskovanju in programiranje vizualizacije sta področji, ki s pomočjo slik predstavita določene dogodke, ki so lahko rezultat računalniških simulacij ali pa procesov in algoritmov. Področje, ki je predstavljeno v naslednjem poglavju, pa je rezultat vpliva vizualizacije na izdelavo in uporabo računalniških orodij.

3.3 Vizualno programiranje

To poglavje predstavlja vizualno programiranje, področje v računalniški vizualizaciji, ki uporablja vizualne metode pri programiranju programske opreme.

Z razvojem grafičnih postaj se je povečal vpliv vizualne tehnologije na okolje jezikov. Priča smo razvoju od popolnoma direktnih prehodov tekstovnih tehnik v odgovarjajočo vizualno tehniko pa do uporabe vizualne tehnike, ki nima nobene direktne paralelne povezave z uporabo tekstovne tehnike. Razpoložljivost vizualne tehnologije vodi k razvoju novih pristopov, ki so vezani na vizualizacijo. V sedemdesetih letih je bilo veliko raziskav tehnologij za razvoj programov usmerjeno v razvoj orodij, ki le malo povezujejo razvoj in vzdrževanje programa. UNIX[1]-ovo razvojno okolje je primer *okolja, ki je podprt s programsko opremo.* Podmnožice takega okolja so v tesni povezavi s programirnim procesom enega programerja. Te podmnožice ali *programska okolja* razlikujejo možnosti za oblikovanje, kodiranje, editiraje, dokumentiranje in razhroščevanje posameznega programerja od možnosti, ki so zahtevane za planiranje, sledenje in upravljanje celotnih projektov programske opreme. Programska okolja lahko vzpodbudijo določene programerske metodologije in določene jezike. Nekatera *okolja jezikov* so tesno povezana okoli enega jezika. Zanimajo nas prav taka, tesno povezana jezikovno določena okolja. Eden prvih okolij jezikov je bil *Interlisp* [33]. *Interlisp* je omogočal osnovne funkcije, ki jih zahtevamo v

[1] UNIX je zaščiteno ime za AT&T Intformation System

okolju jezika: popolno povezano jezikovno določeno okolje s svojim uporabniškim vmesnikom, editorjem, interpreterjem in simboličnim razhroščevalnikom. Vizualizacija je vplivala tudi na razvoj objektno orientiranih jezikov. Danes so najbolj razširjeni *Smalltalk, ObjectiveC* in *C++*. Poleg razvoja novih, objektno orientiranih jezikov, pa naj omenim, da je tudi klasični proceduralni jezik *Pascal* v verziji *Turbo Pascal 5.5* podjetja Borland International Inc. razširil svojo sintakso z objektno orientiranimi strukturami [34]. V času, ko so se okolja jezikov prvič razvila, je bila tehnologija računalnikov v bolj primitivnem stanju. Zato se je do okolja jezikov dostopalo striktno preko tekstovnih ukazov. Vendar je pojav vizualne tehnolologije prinesel dramatične spremembe pri vmesniku med uporabnikom in okoljem jezika.

3.3.1 Vizualni uporabniški vmesniki

Dostop preko več oken (multiple window)

Smalltalk [13] je predstavil ne samo nov jezik, ki je razširil objektno orientiran dostop *Simula67*, ampak tudi nov in visoko vizualen uporabniški vmesnik. *Smalltalk* uporablja tako imenovani vzorec prekrivajočih se oken, ki služijo kot uporabniški vmesnik. Tak vzorec omogoča poljubno velika okna med katerimi lahko poljubno izbiramo. Osnovni vidiki tega vzorca so:

- več uporabniških poslov lahko opazujemo sočasno,
- preklopi med posli se vršijo s pomočjo gumbov,
- nobena informacija se med preklopi ne izgubi,
- prostor na ekranu je porabljen ekonomično,

Tak vzorec vsebuje temelje za uporabniške vmesnike ne samo za okolja jezikov, ampak tudi za sistemske lupine kot je na primer sistemska lupina računalnika *Macintosh* podjetja Apple.

Dostop preko več pogledov (multiple view)

Z razvojem sistema *Pecan* [27] se je uvedel koncept več pogledov. Razlika med večimi okni in večimi pogledi je v tem, da več pogledov omogoča različne predstavitve istih podatkov. Vedno, kadar se katerikoli vidik podatkov spremeni, se simultano spremenijo vsi pogledi in s tem se predstavijo vse spremembe. Ideja večih pogledov je v tem, da lahko uporabnik, s tem, ko se podatki simultano predstavijo na več načinov, izbere tiste poglede, ki so v določenem primeru najbolj uporabni. V *Pecan*-u notranje abstraktno drevo podpira sočasne poglede. Uporabnik nikdar ne vidi tega notranjega abstraktnega drevesa direktno, vendar je njegova informacija na voljo preko različnih sočasnih pogledov. *Software through Pictures* [36] je naslednje grafično orientirano okolje jezika z družino grafičnih editorjev in centralizirano bazo podatkov za informacijo o sistemu med razvojem. Vsak editor podpira določeno metodologijo razvoja ali analize, ki je bila razvita v zadnjih letih. Glavni editorji v *Software through Pictures* so:

- *Dataflow Diagram Editor*, ki podpira strukturirano sistemsko analizo,
- *Entity-Relationship Editor*, ki podpira entitetno odvisnost modelov podatkov,
- *Transition Diagram Editor*, ki podpira USE (user software engeneering) metodologijo,
- *Data Structure Editor*, ki podpira definiranje strukture podatkov,
- *Structure Chart Editor*, ki podpira strukturno oblikovanje.

S.P. Reiss je iz izkušenj s *Pecan* sistemom in drugimi sistemi za zagotavljanje grafičnega pogleda tekstualnih jezikov opazil, da so uporabniki omejeni z enodimenzionalnimi asociacijami, ki so podvržene tekstovnim jezikom [28]. Ugotovil je, da je efektivna uporaba dvodimenzionalnih možnosti grafičnih vpogledov odvisna od jezikov, katerih narava izražanja ni v tekstovni, temveč v grafični obliki. Kot rezultat te ugotovitve je razvil sistem *Garden* [28], ki sprejema tako vizualne kot tudi tekstovne programske objekte. Drugi razvijalci, ki so prišli do enakega zaključka kot Reiss, pa so razvili vizualne jezike (visual languages). *Garden*, prav tako kot *Pecan*, uporablja skupni notranji predstavitveni model—v tem primeru objektno orientirano okolje z vsemi njegovimi povezavami. V novem konceptualnem modelu objekti predstavljajo nove bazične objekte modela. Definirana je nova interpretacija upravljanja z objekti. Vsakemu od teh objektov lahko damo vizualno in tekstovno predstavitev. Za poenostavitev procesa vsebuje *Garden* široko knjižnico baze podatkov, upravljanje procesa, uporabniški vmesnik, dinamični izpis, razhroščevanje ter editiranje možnosti, ki jih ima uporabnik na voljo za definiranje in testiranje novih konceptualnih modelov.

3.3.2 Vizualno editiranje

Sintaksno usmerjeno editiranje

Ena od najbolj prevladujočih tehnik editiranja, ki se pojavi v okolju jezika, je sintaksno usmerjeno editiranje. Editor pozna sintakso jezika in lahko da programerju takojšnje obvestilo o sintaksni napaki ali pa prisili programerja, da uporablja strukturo, ki temelji na sintaksi jezika. Dva sintaksno usmerjena editorja *Cornell Program Synthesizer* [32] editor (okolje jezika *PL/CS*, ki je podmnožica *PL/1*) in *Aloe editor* uporabljen v *Gandalf systemu* [15] uporabljata filozofijo, da vsak del programa temelji na določenem vzorcu in zato mora biti struktura, ki jo naredimo med editiranjem, narejena s pomočjo shranjenega vzorca. Ta filozofija omogoča uporabnikom, da uporablja vizualno hierarhično transverzalno tehniko, ki temelji na strukturi programa.

Specifikacijsko usmerjeno editiranje

Koncept strukturno osnovanega editiranja je lahko razširjena z dodatnimi pravili o strukturi programov. Ta pravila lahko uporabljamo zato, da lahko dovolimo vpis samo takih programov, ki so dokazani s predpisano množico specifikacij. *Use.It* [16] uporablja formalno grafično orientiran pristop k konstrukciji programov. Aplikacije, napisane v *Use.It,* so predstavljene kot binarno drevo, poznano kot kontrolni načrt. Vsako vozlišče drevesa predstavlja funkcijo, ki ima številne vhodne in izhodne objekte. V grafični predstavitvi so vhodni objekti navedeni levo, izhodni pa desno od vozlišča. Listi drevesa so primitivne funkcije. Vozlišča, ki niso listi, se razdelijo v podfunkcije z uporabo kontrolnih struktur (join, include, or). Podobne grafične tehnike se uporabljajo v *PegaSys* [26] sistemu, ki uporablja grafične podobe za formalno predstavitev specifikacije oblike programa. Poudarek v *PegaSys* je na uporabi formalnih grafičnih specifikacij v dokumentaciji in kot preverjanje, ali program uporabnika ustreza specifikacijam.

Prvi pristopi k vizualnem programiranju so bili sestavljeni iz vizualnih editorjev za tradicionalne tekstovne jezike. Pogosto je bila kontrola programa predstavljena z diagrami potekov ali Nassi-Schneidermannovimi diagrami. *Pecan* je vseboval take poglede za predstavitev Pascalskih programov. Kasnejši pristopi so popolnoma odpravili tekstovno verzijo. Kot edina predstavitev programov se uporabljajo diagrami, s pomočjo katerih se program konstruira in editira. Dober primer takega sistema je *Pict/D* [12], ki uporablja običajne nujne jezikovne vzorce, ki zamenjajo vse ključne besede z ikonami. Dobljeni jezik kreira in uporablja diagrame poteka z dodatno možnostjo, da kreira nove ikone, ki predstavljajo podgrafe. Ker je semantika jezika običajna, zahteva programiranje običajne koncepte. Programi, ki jih dobivamo so enake kompleksnosti kot ustrezni tekstovni programi. *Pict/D* se je osredotočil na uporabo vizualnih podob za izboljšanje naše sposobnosti za razumevanje in editiranje programov.

3.3.3 Vizualni jeziki

Vizualno programiranje uporablja bolj vizualno kot tekstovno tehnologijo. Razvoj vizualno programskih jezikov predstavlja nov korak v razvoju vizualnega okolja jezikov. Vizualni jeziki se delijo v dve kategoriji:

- vizualno pretvorjeni jeziki, ki vsebujejo tiste vizualne jezike, ki so nevizualni, imajo pa sposobnost, da se lahko predstavijo vizualno,
- naravno vizualni jeziki, ki želijo razviti nov programski vzorec, katerega naravna predstavitev je vizualna in za katerega ne obstaja noben tekstovni ekvivalent.

Zaradi narave vizualnega programiranja je pogosto težko ločiti vizualne jezike od njihovega okolja (okolja jezika).

Paradigma pretoka podatkov

Vizualni jezik, ki temelji na vzorcu pretoka podatkov lahko smatramo kot vizualno pretvorjeni jezik. Temu bi lahko oporekali, saj vzorec pretvorbe podatkov temelji na diagramih pretoka podatkov, v katerih je program sestavljen v funkcionalne module z povezovanjem poti med vhodom in izhodom. V tekstovnih jezikih, ki temeljijo na tem vzorcu, se diagrami pretoka podatkov normalno narišejo najprej kot del procesa oblikovanja programa in se nato prevedejo v tekstovno sintakso. Vizualni jeziki, ki temeljijo na tem vzorcu enostavno izpustijo pretvorbo v tekst.

Paradigma na osnovi omejitev

Mnogo jezikov, ki temeljijo na tej paradigmi, je dobro prilagojenih vizualizaciji. Prednost vizualne predstavitve te paradigme je skrita v večsmerni naravi prisile. Omejitev lahko vpliva na mnoge spremenljivke, saj se kompliciranе povezave lažje vidijo v diagramski predstavitvi. Primer takega jezika je *ThingLab* [5]. S tem, ko podaja množico omejitev (pravil) in opisuje različne lastnosti in povezave vseh objektov v prostoru problema, določa množico spremenljivk, ki zadovoljuje vsem omejitvam. Z zadostnimi omejitvami lahko povzročimo, da bo množica rešitev poljubno majhna.

Programiranje_z_demonstracijo

To je naravni vizualni proces, za katerega ne obstaja tekstovni ekvivalent. Programira se tako, da grafično upravljamo s podatki na ekranu in s tem demonstriramo računalniku kaj naj program dela. Prednost takega pristopa je vidna—veliko enostavneje je programerju pokazati proces, kot pa tekstovno opisati kako proces

izvesti. *ThingPad* [30] je določljiv, grafičen jezik s svojim okoljem, ki je programiran_z_demonstracijo. Za predstavitev programiranja_z_demonstracijo uporabnik grafično upravlja z diagramskimi predstavitvami strukture podatkov, da bi demonstriral operacije na podatkih. Avtomatična preslikava uporabniškega upravljanja s podatki v kodo *Prolog*-a določi program. *Rehearsal World* [10] je vizualno programirano okolje jezika, ki je bilo razvito za neprogramerje. Je eden novejših okolij, ki podpirajo vizualno programiranje. *Rehearsal World* uporablja gledališke metafore. Osnovne komponente, ki se imenujejo igralci (objekti), sodelujejo drug z drugim na odru (ekran) s pošiljanjem sporočil (signalov). Ekran je oder, na katerem igralci izvajajo akcije, ki jih je naročil uporabnik za določeno igro (program). *Rehearsal World* vsebuje veliko že definiranih igralcev, od katerih vsak razume veliko množico sporočil (signalov). Te igralce lahko kopiramo in jim dodajamo še dodatna znanja. V *Pictorial Transformations (PT)* [20] programiramo tako, da moramo najprej vizualno opisati predstavitev podatkov, nato pa grafično upravljati z njimi in s tem razviti algoritem programa. Objekti v *PT*, imenovani tudi dinamične ikone, sestavljajo pare atribut/vrednost in ikonsko predstavitev funkcij, ki ustvarjajo sliko kot funkcijo novih tipov objektov. Proceduralni jezik *PT* uporablja filmsko izdelane metafore. Programiranje zahteva oblikovanje grafičnih objektov in uporabo teh objektov za demonstracijo delovanja algoritma. "Slika" je kolekcija grafičnih objektov, "film" pa je sekvenca manipulacij s slikami. Programer si izbere začetno sliko, ki jo slikovno transformira. Proces se posname kot en ali več filmov. Z združevanjem ikon objektov v slike in film programer dobi vizualno predstavitev izvajanja programa, ki je rezultat spreminjanja programa.

Oblikovno osnovane paradigme

Oblikovno osnovano programiranje je generalizacija programiranja razpredelnic. Čeprav se v razpredelnicah uporablja tekst v povezavi med podatki in računskimi izrazi, je tekst del oblike, ne pa tekstovno opisanega postopka. Zato so razpredelnice naravno vizualne. Vizualna predstavitev celice matrike dovoli opustitev koncepta spremenljivk, deklaracij in oblik izhodnih podatkov. Prav zato, ker so razpredelnice naravno vizualne, so jeziki za razpredelnice doživeli tak uspeh. *Forms* razširi paradigmo razpredelnice na širšo domeno problemov. Naslanja se na vizualno predstavitev splošnih oblik razpredelnice, da bi minimalizirali koncept jezika. Osnova je forma, ki je kot list papirja, na katerega rišemo celice matrike—objekte. Izraz celice se lahko nanaša na katerikoli objekt v formi, pri pogoju, da ne pride do krožne odvisnosti. Za vsak razvoj forme se vsaka celica izračuna le enkrat. Interakcije in rekurzije lahko omogočimo preko ponavljajočih klicev podform, ki vsakič ustvari nove vsebine podform. Zato množica vseh form in podform, ki jih uporabimo za dani račun, predstavlja zgodovino izračuna in s tem naravno vizualno razhroščevanje.

Področji programiranja vizualizacije in vizualnega programiranja se združujeta z ostalimi medijskimi orodji (zvok, animacija, film) v multimedijo. Programiranje vizualizacije se v multimediji uporablja za predstavitev podatkov, procesov in algoritmov, medtem ko je vizualno programiranje odvisno od programskega orodja v katerem izdelujemo multimedijsko aplikacijo.

4. Poglavje

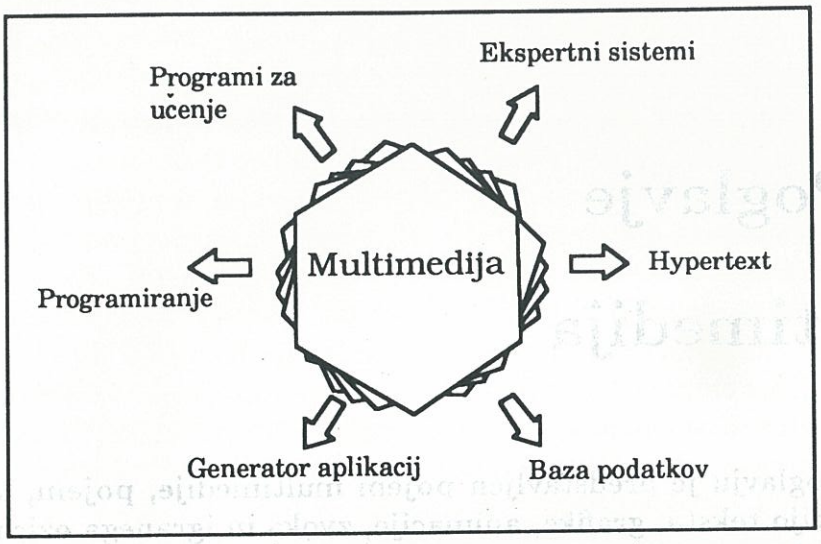
Multimedija

V tem poglavju je predstavljen pojem multimedije, pojem, ki združuje kombinacijo teksta, grafike, animacije, zvoka in igranega oziroma animiranega videa na nelinearen način.

Dandanes želijo učitelji, umetniki, zdravniki, režiserji, plesalci, biologi in širok spekter ljudi drugih poklicev izkoristiti računalnik kot nov medij za izražanje, pridobivanje izkušenj in učenje. Če razširimo vizualne metafore na strukture, ki so primerne za učenje, morajo biti te metafore tako določene, da omogočajo preprostemu, računalniško neizobraženemu uporabniku, da hitro raziskuje in razume nekatere vrste informacij. Z uporabo vizualnih metafor, digitalne audio in video tehnike ter omogočanjem večsmernega raziskovanja s pomočjo računalnika smo dobili nov pojem—multimedijo. Multimedijo lahko uporabimo za toliko nalog in projektov, kolikor obstaja domiselnih ljudi, ki jih znajo ustvariti. Multimedija vsebuje hypermedijo, saj lahko do podatkov dostopamo nelinearno, vendar lahko trdimo, da se ta dva pojma zelo prekrivata in pogosto se multimedija označi kot hypermedija ali interaktivni video. Vendar se v svojem bistvu multimedija vrača h konceptu razvoja človekovega povezovanja z računalnikom. Multimedija je okolje za učenje in dostop do informacije, ki vsebuje idejo n-dimenzij—možnost, da dostopamo do informacije v kateri koli smeri, ki je primerna za informacijo. To okolje dovoli, da se informacija poveže z drugo informacijo, ne glede na to, ali informacija prihaja preko telefona ali pomnilniške lokacije. To okolje dejansko postane informacijsko orodje—mreža znanja, do katerega lahko pridemo na oseben, svojevrsten način, ki je predstavljen tako, da je prilagojen uporabnikovim potrebam.

Multimedijski sistemi so pravzaprav prilagojeni grafičnim vmesnikom. Multimedija razdeli informacijo na posamezne delce. Vsak delec informacije se lahko poveže na mnogo različnih načinov z drugimi delci informacije v sistemu. Delec informacije je omejen z velikostjo ekrana. Povezave med posameznimi delčki so definirane grafično (ikone, gumbi).

Glavni cilji multimedije so:



Slika 4.1: Sistemi, ki prispevajo k n–dimenzionalnosti multimedije

1. Povezovanje in predstavitev informacije na način, ki je lasten človekovemu mišljenju—povezava slik, zvoka, besed ter multidimenzionalne povezave.
2. Zagotoviti primerno okolje za učenje.
3. Omogočiti ljudem, da programirajo brez posebnega znanja o računalništvu in programiranju.

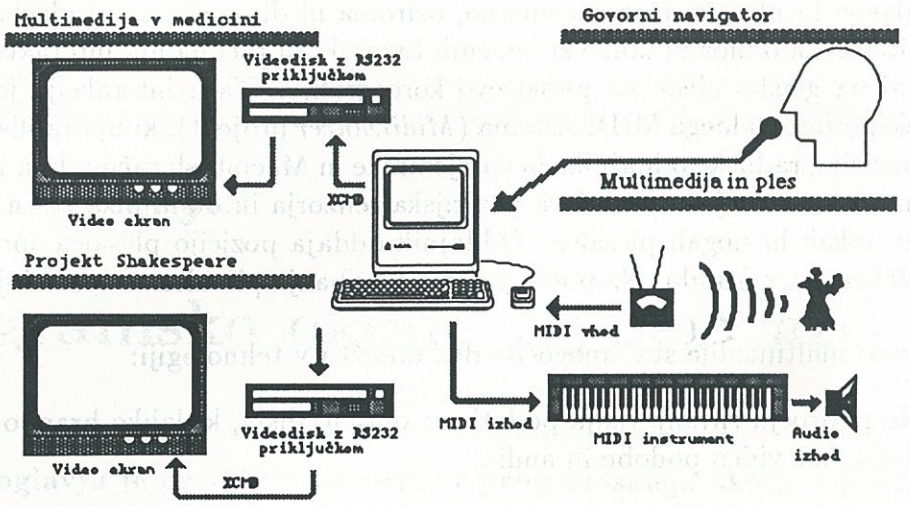
4.1 Uporaba multimedije

- Multimedija v medicini

  Projekt, ki so ga razvili v Standford University Medical School, se imenuje *The Eletrical Cadaver*. To je projekt, kjer so se združile grafika, tekst in slike človeškega telesa. Razvit je bil za pomoč študentom pri študiju anatomije in fiziologije. Študent se lahko uči anatomijo telesa tako, da pregleduje slike kosti, mišic, kit in notranjih organov. Kot dodatek k slikam so podane tekstovne razlage slik. Študent lahko tudi opazuje animacijo delovanja mišic in kit pri gibanju.

- Govorni navigator

  Ena od osnovnih oblik komuniciranja med ljudmi je govor. Razpoznavanje jezika je naloga govornega navigatorja. Navigator je kombinacija strojne in programske opreme in je razvit tako, da so funkcije, ki so najbolj primerne za vodenje preko govora (izbira preko menuja, ukazi, orodja, premikanje kurzorja) tudi realizirane. Delo govornega navigatorja se prične z govorom v



Slika 4.2: Uporaba multimedije

mikrofon (lahko tudi telefon). Signal gre nato skozi analogno/digitalni procesor signalov, kjer se filtrira in kompresira. Tak signal gre nato skozi modul za razpoznavanje. Za razpoznavanje mora uporabnik že predhodno naučiti govorni navigator, kako izgovarja določene ukaze. Navigator ima slovar, ki je hierarhično organiziran in obsega 200 (tudi 1000) uporabnih ukazov. V svojem delovanju navigator ni omejen samo na izvajanje ukazov, temveč ga lahko uporabljamo tudi za predvajanje posnetega sporočila oziroma govora.

- Projekt Shakespeare

  Na Standford University so razvili *Shakespeare Project*, s pomočjo katerega študentje lahko upravljajo z izbranimi sekvencami, primerjajo različne verzije določene scene in izdelajo svojo verzijo epizode na računalniškem odru. Pri tem imajo na voljo na stotine slik iz poprejšnjih oderskih ali filmskih uprizoritev, fotografij, risb kostumov, odrskih rekvizitov, ki jih lahko prilagodijo svoji sceni ter izdelajo svojo študijo karakterjev. Ta projekt omogoča, da lahko filmsko sekvenco izvzamemo iz filma, jo preučimo, analiziramo, spremenimo in vrnemo nazaj v film. S kombinacijo treh tradicionalnih poti učenja drame—branje teksta, opazovanje predstave in produkcije—vzpodbuja *Shakespeare Project* študente, da vidijo tekst kot skelet za široko izbiro možnih interpretacij. Projekt omogoča pet posameznih pristopov: predstavitev, študijo, gledališko igro, pregledovanje in beležko. Trenutna verzija projekta vsebuje glavne scene iz Hamleta, Kralja Leara in Macbetha.

- Multimedija in ples

  Plesalci normalno plešejo po taktih glasbe, ki jih vodi. Partnerstvo med

glasbo in plesalcem je enosmerno, oziroma ni dinamično, saj glasba ukazuje ritem. Ta odnos bi lahko spremenili tako, da bi računalnik interaktivno vplival na glasbo glede na plesalčevo koreografijo. Taka interakcija je bila cilj eksperimentalnega MIDI sistema (*MidiDancer* projekt), ki uporablja senzorje pozicije, radijske oddajnike in sprejemnike in Macintosh računalnik za igranje na sintetizatorje zvoka. Dva pozicijska senzorja in oddajnika sta nameščena na rokah in nogah plesalca. Oddajnik oddaja pozicijo plesalca sprejemniku 50 krat na sekundo. Na podlagi ritma in gibanja plesalca se spreminja glasba.

Razvoj multimedije sta omogočila dva dosežka v tehnologiji:

- Na področju shranjevanja podatkov: optični diski, ki lahko hranijo tako podatke kot video podobe in audio.
- Pojav uporabniškega sistema, ki ni niti urejevalnik teksta, niti razpredelnica, niti baza podatkov, niti igra, čeprav kombinira določene elemente vsakega.

Tak sistem, z imenom *HyperCard*, je bil razvit leta 1987 za računalnik Macintosh podjetja Apple. Ta sistem omogoča kombiniranje teksta, grafike, animacije in zvoka v interaktivni multimediji.

5. Poglavje

Programsko okolje HyperCard

V tem poglavju je predstavljen primer programskega okolja *HyperCard* s programskim jezikom *HyperTalk*, ki je primeren za izdelavo multimedijskih aplikacij. Poglavje je razdeljeno v tri podpoglavja, v katerih se podajajo lastnosti objektno orientiranih jezikov, primerjava objektno orientiranih jezikov s *HyperTalk*-om in predstavitev jezika *HyperTalk* v multimedijskem okolju.

5.1 Objektno orientirani jeziki (OOJ)

To podpoglavje predstavlja lastnosti in prednosti objektno orientiranega jezika, ki je prav zaradi svojih lastnosti primeren za implementacijo vizualnih programskih vmesnikov.

Objektno orientirano programiranje se razlikuje od običajnega tradicionalnega programiranja. V proceduralnem programiranju s pomočjo programskih jezikov *Pascal, C* in drugih podobnih jezikov, se definirajo funkcije in procedure, ki operirajo nad določenim tipom podatkov. Podatki se ločijo od funkcij in procedur, ki operirajo nad njimi. V objektno orientiranem programiranju (OOP), pa so podatki in procedure, ki operirajo nad podatki, zvezani skupaj v strukturo imenovano objekt. V OOP obstaja pet osnovnih elementov: objekti (*objects*), sporočila (*messages*), metode (*methods*), razredi (*classes*) in dediščina (*inheritance*). Vsi elementi so med sabo zelo prepleteni tako v organizaciji kot tudi v delovanju.

5.1.1 Objekti

Gledano z abstraktnega vidika so objekti posamezna programska entiteta, ki povezuje podatke in procedure ali funkcije, ki operirajo nad temi podatki. Z vidika programiranja pa so objekti sistem, ki pošilja in sprejema sporočila. Pri proceduralnem programiranju moramo natančno poznati strukturo podatkov nad katerimi

bo procedura operirala. V OOP pa programerju ni treba poznati strukture podatkov, pomembne so le lastnosti objekta. Če programer že ima definirane določene postopke (procedure, funkcije) nad objekti, ki imajo nekaj enakih lastnosti, kot objekt, ki ga želimo na novo definirati, lahko postopke, ki so že definirani, uporabi in doda samo tiste postopke, ki so specifični za to vrsto objekta. OOP tako omogoča zelo enostavno modifikacijo in vzdrževanje.

5.1.2 Sporočila

Sporočilo v OOP ustreza klicu procedure ali funkcije v proceduralnih jezikih. V OOP je vse narejeno tako, da en objekt pošlje sporočilo drugemu objektu in opazuje njegove reakcije. Model programiranja s sporočili naredi programe pisane za OOP okolje preproste za razumevanje.

5.1.3 Metode

Metoda pravzaprav ustreza funkciji ali proceduri v proceduranih jezikih. Metoda je koda v objektu, ki pove objektu kako naj reagira, ko sprejme sporočilo z nazivom metode. Objekt v OOP ima lahko nič ali zelo veliko število metod, ki jih razume in ki oblikujejo obnašanje in reakcije objekta glede na sporočila drugih objektov.

5.1.4 Razredi

Skupina objektov z množico skupnih karakteristik je razred. Najpomembnejša stvar, ki je skupna objektom in razredu, je način reagiranja na eno ali več sporočil. Če bi pisali enake metode za vsak posamezen objekt, bi izgubili prednosti OOP v ogromni količini kode. Toda, če definiramo razred, bo vsak objekt, ki ga ustvarimo kot element množice, že vedel kako se odzivati na sporočila, ki so vsebovana v razredu. Razred je seveda objekt. Posamezni objekti razreda se imenujejo primerki razreda. Ker je razred tudi objekt, je lahko objekt razred in primerek drugega razreda. Koncept, ki izpostavlja pomembnost razredov, je dedovanje.

5.1.5 Dedovanje

Objekti podedujejo obnašanje od njihovih prednikov preko verige dednosti. Vsi objekti v OOP imajo vsaj enega prednika. Bližje je objekt korenu verige dednosti—objektu, iz katerega se razraščajo ostali objekti in razredi—manj prednikov ima. Ta struktura definira obliko klasifikacijske sheme. Glavna ideja dediščine je v tem, da lahko vsi objekti in podrazredi tega razreda uporabljajo metodo, ki je definirana v tem razredu na enak način. Če določen objekt nima metode, ki jo kliče sporočilo, pošlje sporočilo navzgor po hierarhiji k svojemu predniku, ki sporočilo sprejme, oziroma ga pošlje naprej po verigi dedovanja, če odgovarjajoče metode nima.

5.1.6 Prednosti objektno orientiranega programiranja

Ideje objektno orientiranega programiranja so postale v zadnjem času zelo pomembne. Razlog za to ni le v novem načinu povezovanja programov in podatkov. Lastnost dedovanja v objektno orientiranem programiranju ima številne prednosti v oblikovanju in razvoju programske opreme:

- objektno orientirano programiranje je "naravno"

  Svet, v katerem živimo, je sestavljen iz objektov. Mnogo stvari naredimo tako, da pošljemo sporočilo drugim objektom v našem svetu in se odzivamo na njihova sporočila. Ponavadi delamo tako, da drugim objektom povemo "kaj" želimo od njih, namesto da jim razlagamo "kako" naj to naredijo. "Kako" opisujejo procedure, ki so del proceduralnega programskega modela. "Kaj" pa opisuje nalogo, problem in rešitev v opisnem ali deklarativnem terminu in je del deklarativnega programskega modela, katerega predstavnik je OOP. Svet pač ne deluje proceduralno in zato je bolj enostavno pisati programe, namenjene OOP okolju, kot pa programe za proceduralno orientirano okolje.

- ponovna uporaba programske kode

  Objekt v OOP okoju lahko definiramo tako, da je uporaben v različnih sistemih in ga zato lahko zelo enostavno prestavljamo iz enega sistema v drug sistem. V novem sistemu ni potrebno poskrbeti za nobeno novo deklaracijo ali podatkovno strukturo, niti ni potrebno spreminjati objektov ali procedur. Preprosto vzamemo objekt iz programa A in ga prilepimo v program B ter poženemo. Če ima programer že nekaj izkušenj v OOP konceptih, se najbolj ukvarja z razvojem objektov in orodij, ki se lahko uporabijo v različnih sistemih. Največ časa porabi nato za sestavljanje in vključevanje primernih objektov v nov "svet" ali sistem. Zelo malo časa pa porabi za izdelovanje novih pristopov in modelov objektov in orodij.

- enostavno vzdrževanje

  Če se objekt obnaša na določen način v sistemu A, se bo prav gotovo obnašal enako v sistemu B. Razhroščevanje je efektivno zmanjšano na iskanje sporočil, ki jih pošiljamo neustreznim objektom, sporočil z napačnim številom argumentov in iskanje manjkajočih ali nedefiniranih objektov in metod.

Zdaj, ko smo si ogledali nekaj najbolj značilnih lastnosti in prednosti objektno orientiranih jezikov, pa si oglejmo, koliko lastnosti objektno orientiranih jezikov je vsebovanih v programskem okolju *HyperCard* s programskim jezikom *HyperTalk* in kako so te lastnosti implementirane.

5.2 OOP in *HyperCard* ter *HyperTalk*

V tem podpoglavju je podana primerjava lastnosti programskega jezika *HyperTalk* z lastnostmi objektno orientiranih jezikov in implementacija lastnosti objektno orientiranih jezikov v *HyperTalk*-u.

HyperTalk ni OOP v pravem pomenu besede, saj ne vsebuje nekaj osnovnih lastnosti pravega OOP. Obstaja nekaj podobnosti med *HyperTalk*-om in pravimi OOP sistemi, čeprav te podobnosti ne veljajo povsod v arhitekturi *HyperTalk*-a.

5.2.1 Objekti in *HyperTalk*

V *HyperTalk*-u obstaja pet tipov objektov: skladi (*stacks*), ozadja (*backgrounds*), kartice (*cards*), gumbi (*buttons*) in polja (*fields*). Tako kot objekti v OOP lahko tudi ti objekti pošiljajo in sprejemajo sporočila. Vsak tip objekta lahko povežemo s scenarijem (*script*), ki vsebuje igralce (*handlers*), ki ustrezajo metodam v OOP. Tako so objekti in programska koda, ki omogoča odzivanje na sporočila, povezana skupaj tako kot objekti v OOP.

5.2.2 Sporočila in *HyperTalk*

HyperTalk uporablja enak termin za opisovanje komunikacije med objekti kot objektno orientirani jeziki. *HyperTalk* vsebuje sistemska sporočila, ki so rezultat dogodkov, povzročenih s strani uporabnika sklada. Vsak tip sporočila se lahko naslovi enemu ali večim tipom objektov s pomočjo *HyperCard*-ovega usmerjanja. Konceptualno gledano je sklad sistem, katerega naloga je, da pazi na dogodke, na katere se mora odzvati. Ko se dogodek zgodi, mora ustvariti sistemsko sporočilo in ga poslati objektu, kateremu je sporočilo namenjeno. Ta objekt se odzove na način, ki je definiran v vlogi igralca za to sporočilo v scenariju objekta. Ta način delovanja je močno podoben delovanju OOP.

5.2.3 Metode in *HyperTalk*

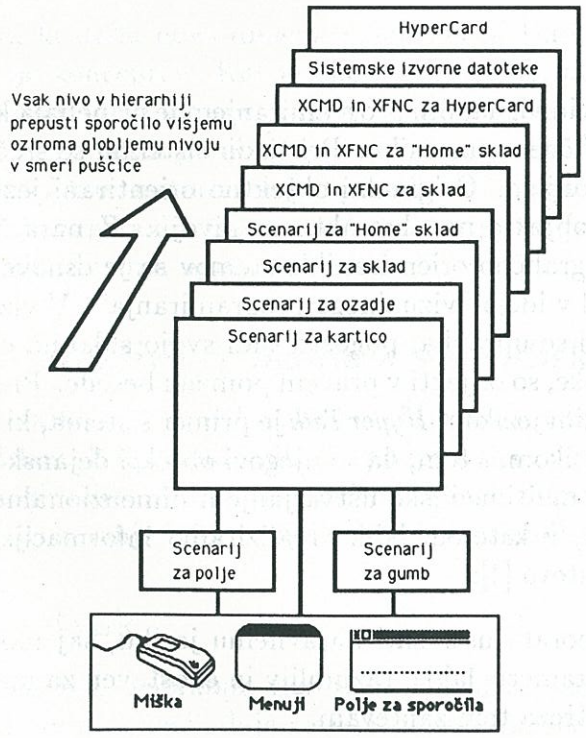
Vsak objekt ima scenarij in v vsakem scenariju je lahko eden ali več igralcev. Igralci ustrezajo metodam v OOP. Igralec je povezan z vsakim tipom sporočila, ki ga objekt lahko sprejme. Obstajata dva tipa igralcev v scenariju *HyperTalk*-a: funkcijski igralci in "on" igralci. "On" igralci so tako imenovani dogodkovni igralci, saj so sproženi z nekim dogodkom v sistemu. Objekti v *HyperCard*-u pa imajo tudi določene lastnosti (properties), od katerih imajo nekatere lastnosti sporočil.

5.2.4 Razredi in *HyperTalk*

Med *HyperTalk*-om in OOP ne obstaja analogija v konceptu razredov. Vzrok leži v tem, da *HyperTalk* ne podpira dedovanja v OOP smislu. Tako na primer ne obstaja razred gumbov, kateremu bi pripadali vsi gumbi, ki bi imeli tudi individualne lastnosti gumba, čeprav vsi gumbi delujejo podobno (na pritisk sprožijo določeno operacijo). Koncept ozadij kartic najbolj spominja na OOP razrede. Vse kartice v skladu z istim ozadjem imajo mnogo skupnih karakteristik. Izgledajo enako in gumbi na ozadju izgledajo in delujejo enako za vse kartice. Vsaka kartica pa ima lahko še svoje gumbe, polja in slike. Ko kreiramo sklad grupiramo kartice s podobnimi funkcijami v ozadja v skladu. Kompleksni skladi imajo skoraj vedno več kot samo eno ozadje. Analogija z OOP razredi pa ni popolna, saj lahko postavimo specifične kartice v isto ozadje, namesto da bi jih generirali s pomočjo programa kot posledico njihove funkcionalne podobnosti ali kot rezultat ukaza.

5.2.5 Dedovanje v *HyperTalk*-u

V *HyperTalk*-u ne obstaja pravo dedovanje. Sporočila prehajajo skozi določeno hierarhijo, ki ima nekaj karakteristik OOP strukture dedovanja.



Slika 5.1: Hierarhija prepuščanja sporočil v *HyperCard*-u.

Hierarhija poteka od gumba ali polja, kjer se zgodijo akcije, ki sprožijo dogodek,

do ozadij, kartic in sklada, do "Home" sklada in končno do samega *HyperCard*-a. Ko se sproži sporočilo, aktivirano s pritiskom miške na gumb, se spusti skozi hierarhijo dokler se ne zgodi ena izmed naslednjih dveh stvari: odkrije se igralec z enakim imenom, kot je sporočilo, ki odigra potrebno vlogo ali pa se detektira vrh hierarhije, ne da bi odkrili igralca s tem imenom. Tako obnašanje je podobno procesiranju sporočil v OOP sistemu. To v obratni smeri ne velja. Če ima nek gumb na kartici sposobnost, da se odzove na določen tip sporočila, ni nujno, da imajo tudi drugi gumbi to sposobnost. Enako velja tudi za kartice in ozadja.

Poleg vseh zgoraj naštetih lastnosti omogoča *HyperCard* uporabo zunanjih funkcij in komand (*XCFN* in *XCMD*), ki jih napišemo v programskem jeziku *Pascal* ali *C*. S tem *HyperCard* omogoča funkcionalno razširitev na področja, ki jih *HyperCard* ne pokriva in s tem omogoča razvijalcem sistema nove ter individualne pristope.

Sedaj pa poglejmo, kako se programsko okolje *HyperCard* s programskim jezikom *HyperTalk* vklaplja v koncept izdelave multimedijskih aplikacij.

5.3 Multimedija in *HyperCard* ter *HyperTalk*

V tem podpoglavju so predstavljene zahteve za jezik, v katerem se lahko implementirajo informacijske multimedijske aplikacije. Podane so tudi lastnosti operacij, ki so zahtevane v multimedijskem okolju in njihova realizacija v *HyperTalk*-u.

Konvergenca med multimedijo in vizualnim programiranjem je pripeljala k vizualizaciji sistemov osnovanih na grafičnem vmesniku. Pri takih sistemih se srečamo z objektno orientiranem programiranjem. Originalni objektno orientirani jeziki kot so *Smalltalk* in *Simula* kreirajo objekte na abstraktnem nivoju. Z naraščanjem popularnosti in razpoložljivostjo grafično orientiranih sistemov se je osnoven koncept objektne orientiranosti razvil v idejo "vizualnega programiranja". V vizualnih sistemih imajo objekti, s katerimi se upravlja, pogosto tudi svojo sliko na ekranu. Strukture, ki implementirajo te slike, so objekti v pravem pomenu besede. Programsko okolje *HyperCard* s programskim jezikom *HyperTalk* je primer sistema, ki je zelo podoben objektno orientiranim jezikom, s tem, da so njegovi objekti dejansko vidni na ekranu in je sistem prirejen za multimedijsko ustvarjanje n-dimenzionalnega informacijskega prostora. Za jezik, v katerem lahko realiziramo informacijske aplikacije se zahtevajo naslednje zahteve [4]:

- Terminologija in sintaksa morata ustrezati naravnemu jeziku, saj mora biti jezik, ki je razvit za ne-programerje lahko razumljiv in enostaven za uporabo. Angleški jezik kot model ustreza tem zahtevam.

- Zaradi razširjenosti mora vsebovati tudi učinkovite kontrolne strukture kot jih ima na primer Pascal. Vsebovati mora osnovne elemente katerega koli strukturnega jezika za gradnjo učinkovitih aplikacij.

- Vsebovati mora širok spekter vgrajenih ukazov, ki so združene z namenom kreiranja informacijskih aplikacij. Scenariji aplikacij naj bodo čim bolj kratki in zgoščeni.

HyperCard je sistem, ki ustreza vsem zgoraj naštetim pogojem in podpira operacije, ki se zahtevajo v multimedijskem sistemu. Te operacije so naslednje: navigacija informacij, animacija, razširitev okolja z videodiskom ter zvok.

5.3.1 Navigacija

V grafično orientiranih multimedijskih sistemih se lahko povezave informacij definirajo z uporabo gumbov. V enostavnih sistemih služijo gumbi za enostavno navigacijo med karticami. V sistemu, ki podpira n-dimenzionalen informacijski prostor, pa gumbi vpeljejo različne akcije: povezovanje z drugimi delčki informacij, povezovanje animacijskih sekvenc, kontrolo videodiska, igranje glasbenih sekvenc in podobno. V *HyperCard*-u se lahko definirajo informacijske poti linearno, hierarhično ali popolnoma naključno. Tako lahko avtor določi povsem poljubno sprehajanje po vozliščih informacijske mreže.

5.3.2 Animacija

Tehnika, ki doda novo dimenzijo informacijskem okolju, je uporaba animacije za ilustracijo konceptov. Kot je bilo že omenjeno, podaja animacija dodatno dimenzijo pri podajanju informacije, saj se skozi animacijo razbere določena informacija, ki se iz vsake slike posebej ne razbere. Pomembna je tudi uporaba animacije za vizualne efekte: način vizualne predstavitve prehoda med posameznimi karticami. Animacija zagotavlja tudi metodo za poenostavljanje poti skozi informacije za zahteve individualnih uporabnikov.

5.3.3 Razširitev okolja z videodiskom

Ena od najbolj zanimivih stvari za oblikovalce v multimedijskem okolju je uporaba videodiskov. Z uporabo jezika, ki podpira tako periferno napravo, lahko oblikovalci sistema združijo informacije z videodiska v informacijsko aplikacijo tako enostavno, kot je združevanje informacij iz ostalih perifernih naprav.

5.3.4 Zvok

Naslednja zelo uporabna razširitev informacijskega okolja je zvok. Programski jezik, ki podpira zvok, daje razvijalcu informacijske aplikacije možnost za uporabo zvoka na različne načine: kot opozorilna naprava, ki poveča zanimivost animacijske sekvence, ali pa naprava za zagotavljanje informacij preko simulacije človeškega govora.

Za predstavitev delovanja vseh zgoraj naštetih lastnosti sem izdelala aplikacijo v programskem okolju *HyperCard* s programskim jezikom *HyperTalk*, ki služi kot primer informacijske aplikacije in ni bila izdelana za določenega uporabnika.

5.4 Primer *HyperCard* aplikacije

V tem poglavju bom predstavila aplikacijo izdelano v programskem okolju *HyperCard* s programskim jezikom *HyperTalk*, ki služi kot primer informacijske aplikacije v hyper/multi-medijskem okolju.

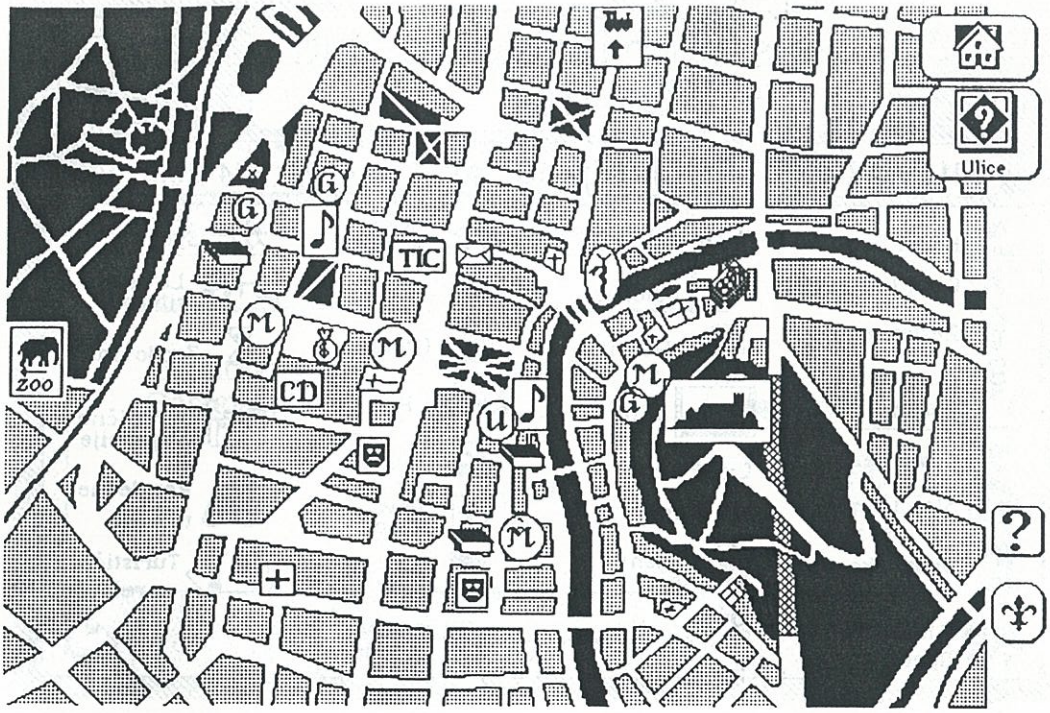
Na računalniku Macintosh podjetja Apple sem v programskem okolju *HyperCard* v programskem jeziku *HyperTalk* razvila turistično informacijsko aplikacijo *Ljubljana*.

5.4.1 Vsebinska predstavitev aplikacije *Ljubljana*

Aplikacija je izdelana na podlagi informacij turistično informacijskega centra v Ljubljani ter turističnega vodnika [24]. V aplikaciji so izdelana orodja in primeri za dostop do turističnih informacij. Kot prva, vstopna kartica aplikacije se pokaže zemljevid Ljubljane (Slika 5.2), na katerem so s posebnimi znaki označene turistične zanimivosti in drugi objekti (križ za cerkev, riba za ribarnico, košara za tržnico, maska za gledališče in podobno). Uporabnik lahko s pritiskom na miško nad izbrano ikono pride do podatkov o objektu, za katerega je ikona definirana. Nekatere ikone imajo definirane različne operacije glede na enojni ali dvojni pritisk gumba na miški. Tako so na zemljevidu Ljubljane označene galerije, muzeji, knjižnice, cerkve, Cankarjev dom, drama, opera, Križanke, filharmonija, univerza, tržnica, ribarnica, lekarna, grad, pošta, banka, zdravstveni dom, Turistično informacijski center, živalski vrt in železniška postaja.

Na tej kartici pa obstaja tudi posebna ikona preko katere se lahko pride do informacije o ulicah v Ljubljani. S pritiskom na miško nad to ikono se pokaže prazen zemljevid Ljubljane in v levem zgornjem kotu je polje v katerega se izpiše ime ulice. Sedaj lahko uporabnik pritiska ali pa drži pritisnjen gumb medtem, ko premika miško nad pikami, ki so narisane v vsaki ulici, in v polju za ime ulice se izpiše ime ulice. Za vsako ulico lahko pridemo do podatkov preko ikone na tej kartici. Če so v aplikaciji na voljo slike objektov v tej ulici, lahko uporabnik pride tudi do teh slik.

Preko ikone za železniško postajo se pride do informacije o železniški postaji (telefon, informacije, odhodi in prihodi vlakov), avtobusni postaji (telefon, informacije, prodaja kart, avtobusni vozni red), prodaji letalskih vozovnic pri JAT-u, Adrii Airways in turističnih agencijah, taksijih, najemu avtomobilov (seznam agencij) ter mestnem prometu.



Slika 5.2: Vstopna kartica v aplikaciji *Ljubljana*.

Preko ikone turistično informacijskega centra pa se pride do informacije o hotelih, restavracijah, gostilnah, sporedu ljubljanskih kinematografov, deviznem tečaju, avto servisih, živalskem vrtu, koledarju prireditev v Ljubljani, zgodovini Ljubljane, turističnih agencijah in turističnih sprehodih po Ljubljani.

Poleg pravkar naštetih možnosti pa je možno z nekaj popravki izdelati tudi dostop do ostalih podatkov kot so na primer banke (zemljevid, obratovalni čas, povezava na devizni tečaj), diskoteke, prireditve v drami, operi in Cankarjevem domu, lov in ribolov, šport (koledar prireditev, možnosti rekreativnega športa urejene po športnih panogah, povezava na lov in ribolov) in tako dalje.

Pri turističnih sprehodih po Ljubljani je izdelan le eden turistični sprehod po mestu, ki vključuje tudi slike nekaterih najbolj zanimivih objektov (slike so iz turističnega vodnika [24]). Prav tako je s pomočjo animacije prikazana pot turističnega sprehoda.

Aplikacija seveda omogoča poljuben sprehod med zgoraj naštetimi podatki in iz vseh kartic je možna takojšna vrnitev na vstopno točko v aplikaciji—prvi zemljevid Ljubljane. Od tu lahko zapustimo aplikacijo in se vrnemo v osnovni "Home" sklad.

5.4.2 Primer sprehoda med karticami

Kot prva, vstopna točka v aplikacijo *Ljubljana* se pokaže kartica na sliki 5.2. Miško pritisnemo nad ikono turistično informacijskga centra (TIC). Na ekranu se prikaže slika 5.3.

Slika 5.3: Kartica turistično informacijskega centra.

Na nej so zbrane ikone, preko katerih lahko pridemo do podrobnejših informacijah o področju, ki ga ikona predstavlja. Izberemo ikono o turističnih agencijah.

Pokaže se kartica, ki je na sliki 5.4. Vidimo lahko, da je na tej kartici več gumbov. Tako lahko s pomočjo puščic levo in desno izbiramo prehode med podatki o turističnih agencijah. Ikona za Ljubljano nas vrne na prvotno, vstopno kartico, zavita puščica v levo pa nazaj na kartico turistično informacijskega centra. Izberemo gumb "Lokacija", ki nam pokaže lokacijo turistične agencije na zemljevidu Ljubljane (slika 5.5).

Tu lahko poljubno premikamo miško po zemljevidu Ljubljane. V zgornjem kotu se bo izpisal naziv ullice, nad katero smo pritisnili miško (slika 5.6).

Če nas zanima kakšni so podatki o ulici, katere ime je izpisano v pravkar omenjenem polju, pritisnemo gumb "Podatki o ulici". Pokaže se slika 5.7.

Če pritisnemo gumb, ki je v desnem zgornjem kotu, se na ekranu pokaže polje v katerem je opisana zgodovina ulice (slika 5.8).

Na tej kartici pa lahko pogledamo tudi sliko objekta oziroma stavbe, ki je v tej ulici. Pritisnemo na gumb, ki predstavlja fotoaparat. Na ekranu se pokaže kartica, ki je na sliki 5.9.

Na tej kartici lahko, s pritiskom na ikoni Ljubljane, pridemo na vstopno kartico Ljubljane. S pomočjo puščic levo in desno pa se premikamo med slikami, ki so vsebovane v aplikaciji. S puščico levo pridemo do slike mestne hiše (slika 5.10).

Na tej kartici je definiranih več nevidnih gumbov. Če pritisnete na miško nad Robbovim vodnjakom, dobite podatke o njegovi zgodovini. Enako velja tudi za

Slika 5.4: Kartica iz "ozadja" turističnih agencij.

mestno hišo in stolnico. Pritisnimo gumb nad stolnico. Na ekranu se pokaže slika stolnice (slika 5.11).

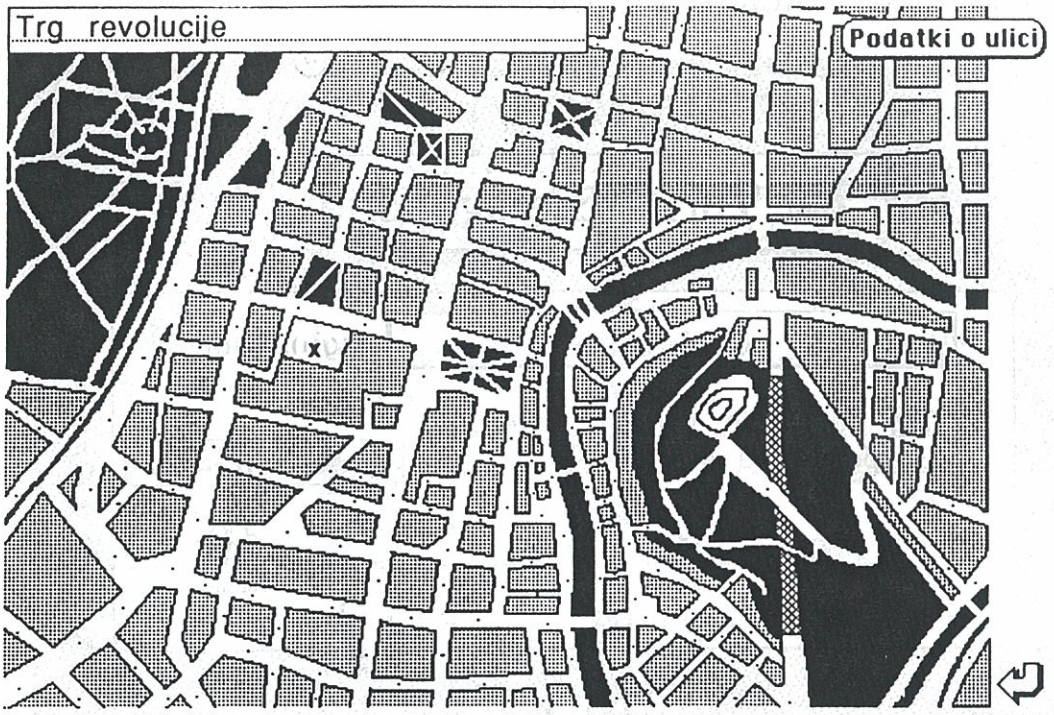
Tu lahko, prav tako kot na prejšnji kartici, pritisnemo miško nad sliko stolnice. Na ekranu se nam pokaže kartica, na kateri so podatki o zgodovini stolnice (slika 5.12).

Ko se vrnemo na kartico, na kateri je slika stolnice, pritisnemo gumb, ki predstavlja turistični sprehod po Ljubljani. Ta gumb ima definirani dve operaciji, ki sta odvisni od vstopne točke v kartico. Do kartice lahko pridemo preko turističnega sprehoda ali pa preko ostalih gumbov. Pri turističnem sprehodu nam ta gumb pomeni nadaljevanje turističnega sprehoda, v vseh ostalih primerih pa nam predstavlja vstopno točko v kartico turističnih sprehodov (slika 5.13).

Sedaj pritisnemo gumb Ljubljana in se vrnemo na prvo kartico, od tu pa z gumbom, ki predstavlja hišo, v "Home" sklad.

5.4.3 Organizacijska predstavitev aplikacije

Vse kartice aplikacije so zbrane v enem samem skladu. Razlog za to je hitrejši dostop do vseh informacij, saj prehod skozi kartice ni linearen in bi v nasprotnem primeru lahko prišlo do menjave sklada ob vsaki zamenjavi kartice, kar bi upočasnilo dostop do informacij. Kartice se kljub temu, da so v istem skladu, ločijo v ozadju in uporabnik ni prav nič prikrajšan pri pomenskem grupiranju kartic in s tem pri informiranju.

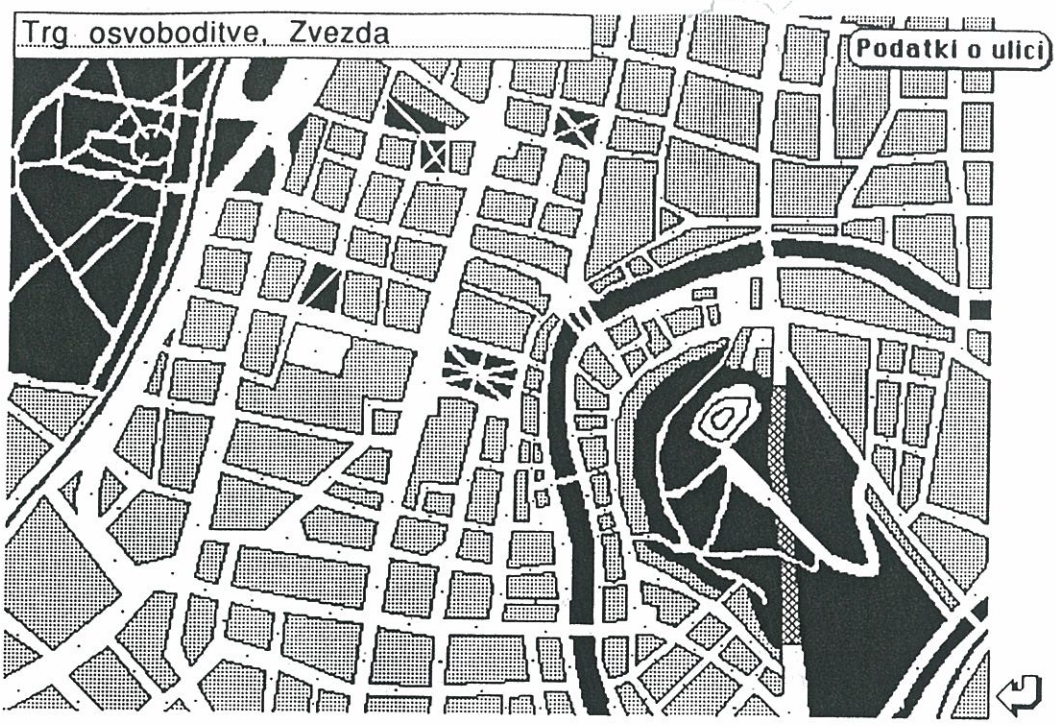


Slika 5.5: Lokacija turistične agencije na zemljevidu Ljubljane.

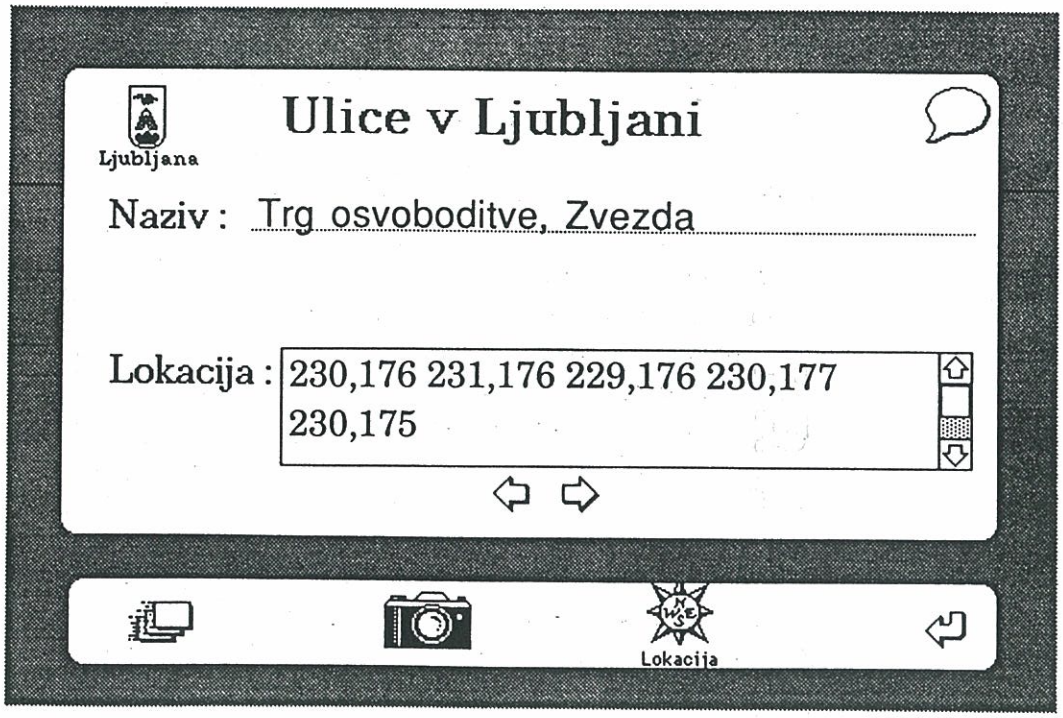
Vsi prehodi med karticami se vršijo izključno preko gumbov nad ikonami. Za nekatere ikone se loči različna operacija glede na število pritiskov na miško (eden ali dva).

V aplikaciji je tudi veliko slik. Slike turističnih objektov so skanirane s pomočjo programa *MacImage*. Slike so se prenesle v program *MacDraft*, kjer sem jim določila velikost in jih nato vključila v aplikacijo. Vse ostale slike so narejene s pomočjo grafičnega orodja v *HyperCard*-u.

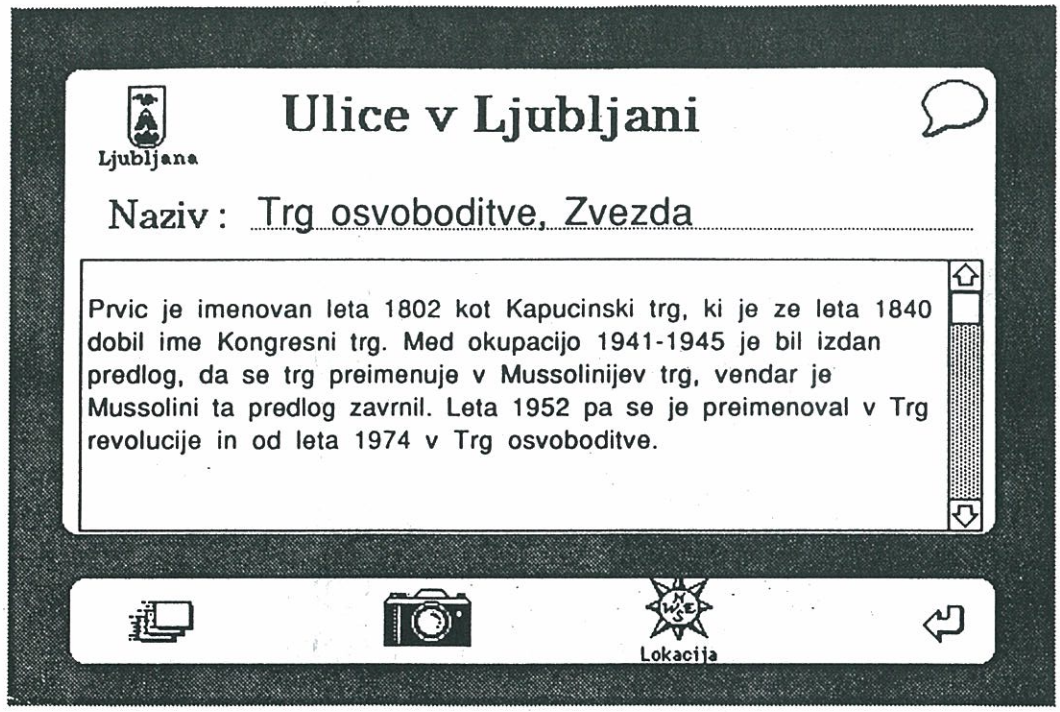
V aplikaciji sem uporabljala slike in obliko črk za ikone, ki so nam znane iz vsakdanjega življenja. Tako naj kot primer omenim znak za avto Mercedes (pozna ga večina ljudi), ki sem ga uporabila pri avto servisih. Pri izdelavi in oblikovanju sklada in scenarijev sem se držala navodil, ki so predlagana v priročnikih za *HyperCard*[14,31].



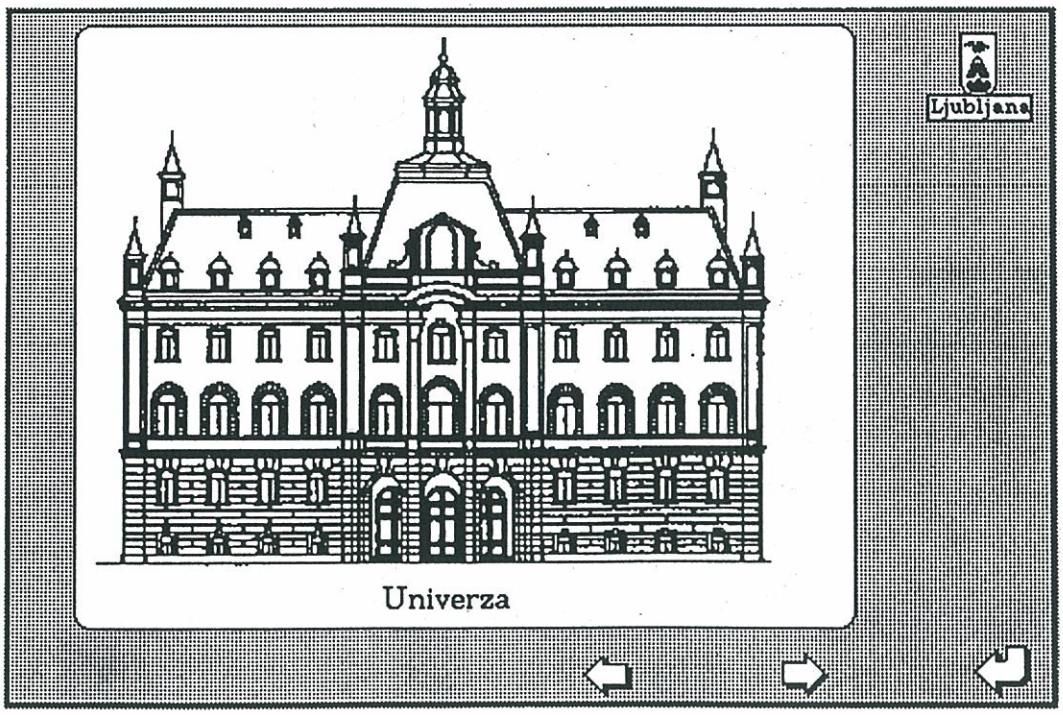
Slika 5.6: Zemljevid Ljubljane na katerem lahko iščemo nazive ulic.



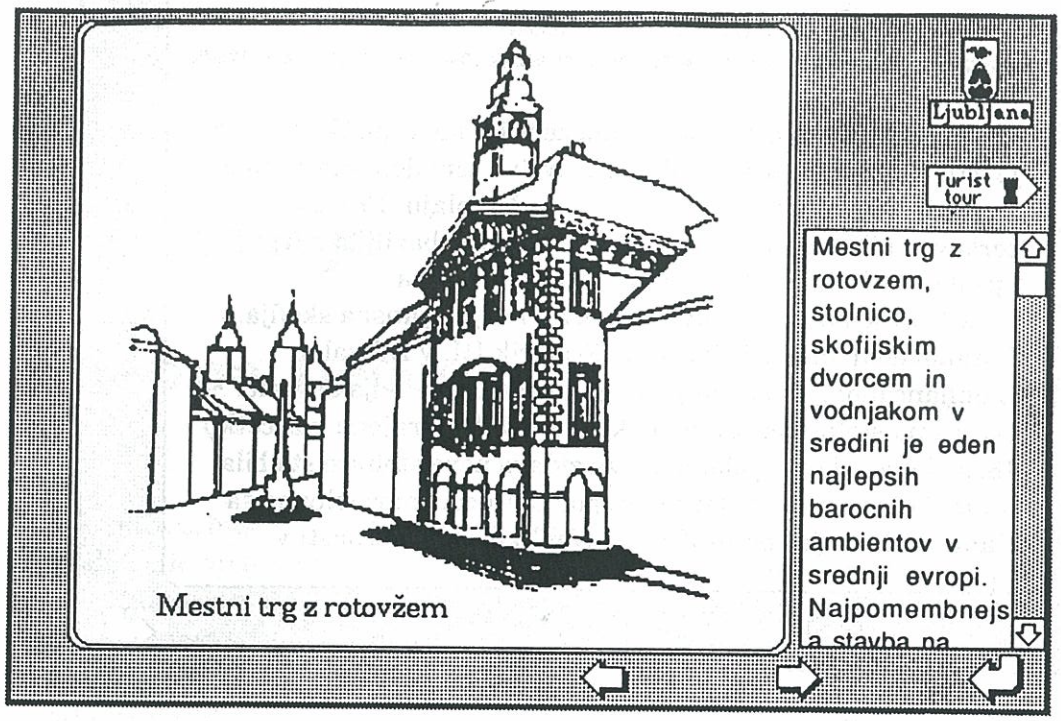
Slika 5.7: Kartica ulice, katere ime je bilo izpisano v polju naziva ulice.



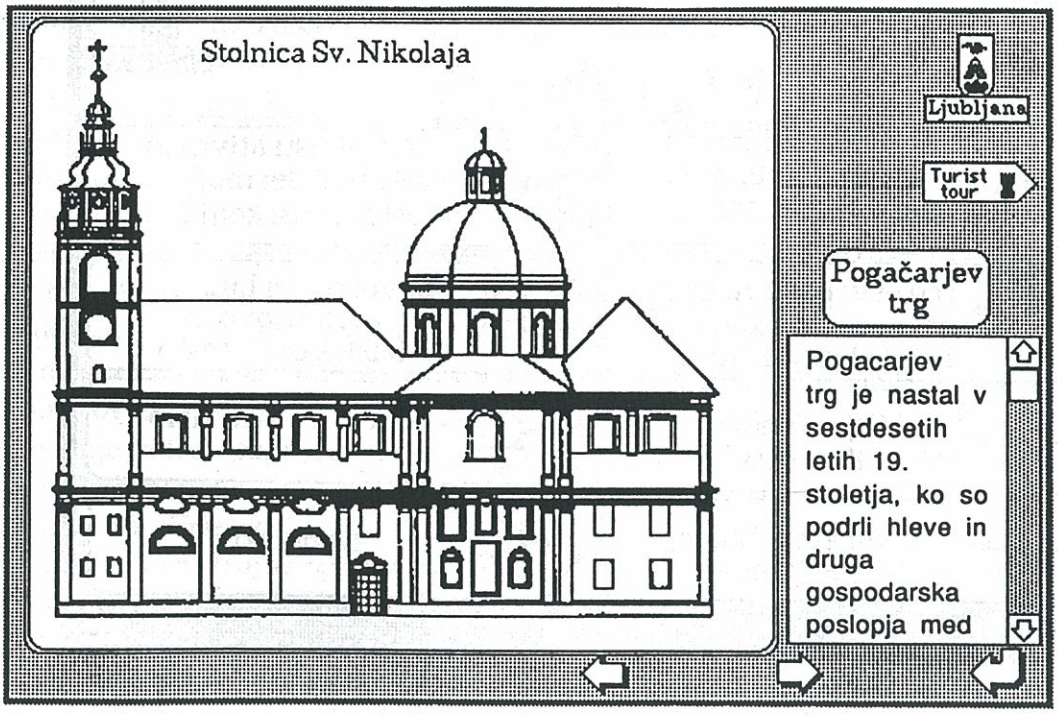
Slika 5.8: Kartica ulice s podatki o zgodovini ulice.



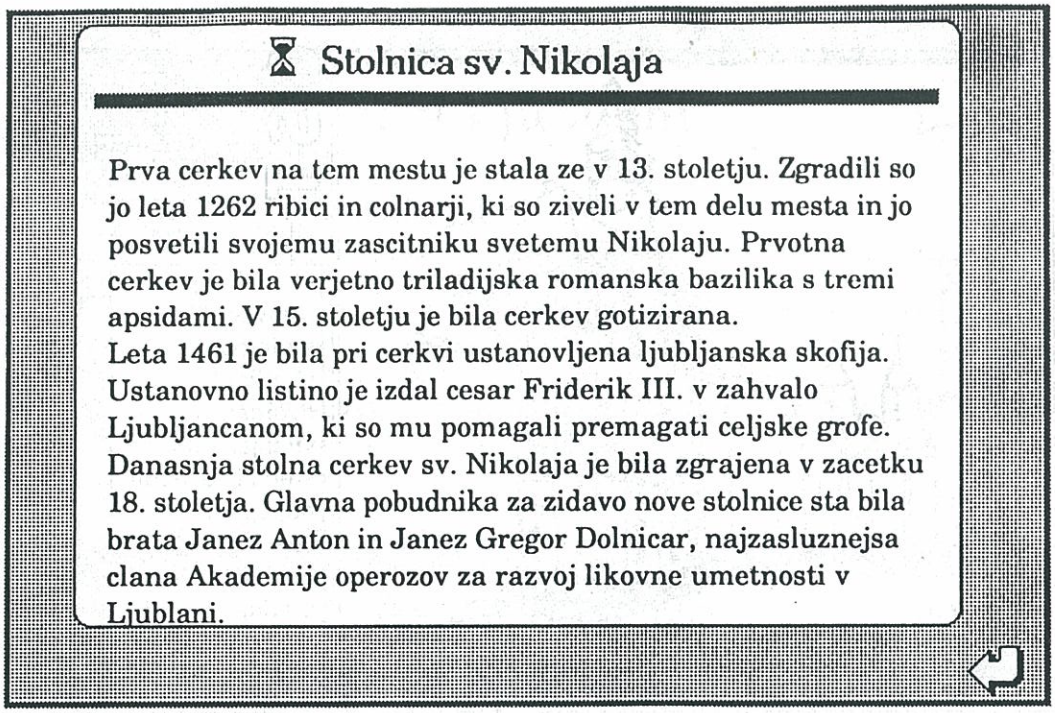
Slika 5.9: Univerza.



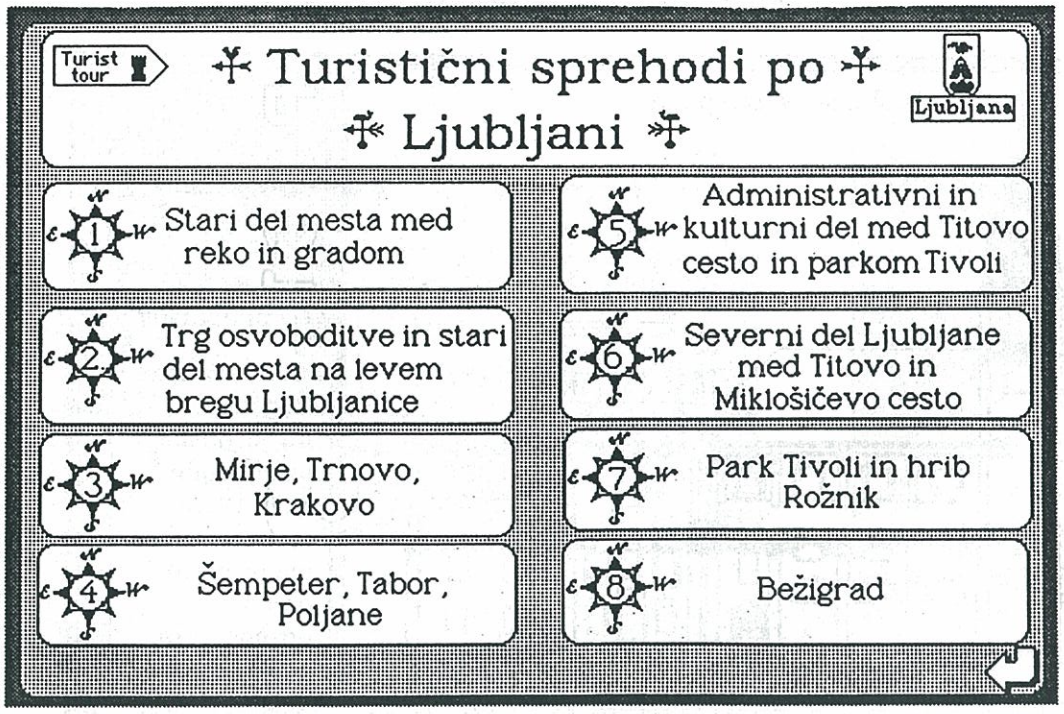
Slika 5.10: Mestni trg z rotovžem.



Slika 5.11: Stolnica sv. Nikolaja.



Slika 5.12: Kartica, ki vsebuje podatke o zgodovini stolnice.



Slika 5.13: Kartica turističnih sprehodov po Ljubljani.

6. Poglavje

Sklepna beseda

Razvoj računalništva gre v smeri enostavnejšega dela z računalnikom. Razvoj računalniških vmesnikov je potekal od linearno, ukazno osnovanega sistema do metaforičnega in danes zvokovno razpoznavnega sistema ter sistema, ki sledi očesu in gibanju telesa (projekt MidiDancer). Računalniški vmesnik bi moral pustiti, da uporabniku ni treba razmišljati o tehnologiji, ki jo uporablja. Vmesni medij bi moral biti neviden gostitelj na računalniku, ki upravlja s sporočili uporabnika.

Ena od možnosti za definiranje takšnih vmesnikov se kaže v kombinaciji multimedije in hypermedije. Nelinearno, poljubno organizirani vzorci informacij, predstavljeni s sliko, tekstom, zvokom in animacijo, so zelo primerni za vse vrste računalniških vmesnikov. Tako ni presenetljiv dejstvo, da vedno več računalnikov uporablja take pristope tudi na nivoju operacijskega sistema (Apple Macintosh).

Hypermedijski sistemi se bodo razvijali v smeri idealnega okolja za komunikacijo in upravljanje z informacijami. V prihodnosti se bodo za kreiranje informacijskega prostora verjetno raje uporabljali ti sistemi namesto linearno oblikovanih dostopov do informacij. Ta nova okolja bodo vsebovala široko paleto programskih orodij kot so sistemi za učenje, upravljalci baz podatkov in tradicionalna programska orodja.

Multimedija se uveljavlja predvsem pri namiznih računalnikih.V lanskem letu je bil prav multimedija pojem, o katerem se je največ govorilo v zvezi z osebnimi namiznimi računalniki. Veliko število člankov o multimediji v strokovnem tisku in Apple–ov zelo popularen *"Knowledge Navigator"*, namizni informacijski sistem, ki vključuje video, dostop do podatkov, elektronsko pošto in telekomunikacije, nam kaže, da postaja multimedija zanimiva tako za znanstvenike kot za strokovnjake v industriji in širše občinstvo.

Kar se tiče strojne opreme, pa je dejanskih produktov pravzaprav malo. Aapps Corp., Sunnyvale, California, je predstavil video ploščo, ki prikazuje televizijski ekran na Macintosh II ekranu. Intel Corp. nadaljuje z izboljševanjem DVI (Digital Video Interactive) tehnologije, vendar novi produkti še niso prišli na trg navkljub govoricam o tem, da bo IBM vgradil DVI v vstopni nivo *PS/2* sistema. Še najbolj popolna multimedijska računalnika sta Macintosh, podjetja Apple in FM Towns podjetja Fujitsu Ltd., katerega proizvajalec imenuje kar "hypermedijski" PC [35].

Multimedija je trenutno "vroča" tema. Nekateri napovedujejo multimediji ogromen trg—11.4 bilionov dolarjev za strojno opremo v letu 1993 [35] oziroma 16.6 bilionov dolarjev v letu 1994 [39]. Nekateri menijo, da bo prav multimedija povzročila tehnološki napredek namiznih računalnikov. Seveda pa bo poleg strojne opreme potrebna tudi programska oprema, ki bo vključevala audio in video.

Primer takega programskega orodja je *HyperCard*. *HyperCard* s programskim jezikom *HyperTalk* ni univerzalen jezik, ga je pa mogoče uspešno razširiti z vključevanjem zunanjih ukazov in funkcij (XCMD in XFNC) tako, da se lahko uporablja za vse aplikacije, katerih funkcije je mogoče realizirati v programskem jeziku *Pascal* ali *C* in *HyperCard* še vedno ohrani svoj hyper/multi-medijski koncept delovanja.

Nakazuje se tudi novo področje v smeri multimedijskih aplikacij in sicer oblikovanje modelov poljubnega okolja. V povezavi z izpopolnjemi senzorji bi lahko uporabnik zaigral v ekipi svoje najljubše košarkarske ekipe ali pa se sprehodil po Luni oziroma vstopil v popolnoma nov abstrakten domišlijski svet.

Ob koncu naj še omenim, da je v mesecu marcu tega leta (1990) začela izhajati nova revija, ki je namenjena raziskovalcem in razvijalcem vizualnih jezikov ter njihovi implementaciji v računalništvu [22]. Revija vsebuje članke o vizualnih jezikih, vizualnem programiranju, oblikovanju vmesnikov, multimedijskih komunikacijah, kognitivnem vidiku sistemov uporabnik-računalnik, bioloških vizualnih sistemih in modelih, vizualizaciji računskih procesov, predstavitvi vizualnih informacijskih procesov, računalniških animacijah in podobno.

Vidimo lahko, da se multimediji in hypermediji posveča vedno več pozornosti tako v znanstveno raziskovalnih kot tudi v poslovnih krogih.

A. Dodatek

Slovar novih pojmov

abstrakcijska funkcija $\mathcal{A}$ Funkcija, ki preslika računska stanja v objekte, ki so grafični objekti razširjeni s parametri o poziciji, velikosti, orientaciji in podobno.

dostop preko večih oken (*multiple window*) Metoda vizualnega uporabniškega vmesnika (uporabljena v *Smalltalk, X-windows, Macintosh Finder, Microsoft Windows*), kjer lahko uporabnik izbira med poljubno velikimi okni z namenom sočasnega opazovanja uporabniških poslov brez izgube informacije ob preklopih med okni.

dostop preko več pogledov (*multiple view*) Metoda vizualnega uporabniškega vmesnika (uporabljena v sistemu *Pecan*), kjer je v vsakem oknu različna predstavitev istih podatkov, ki se sočasno spreminjajo in uporabnik lahko izbere tisto predstavitev podatkov, ki je v danem trenutku najbolj uporabna.

hypermedia Medij v katerem so podatki organizirani v nelinearnem modelu ali v plasteh z namenom predstavitve spremenljivih nivojev podatkov.

interpretacijska funkcija $\mathcal{I}$ Funkcija, ki predstavi simbole s sliko.

kognitivnost Poznavanje, zavedanje, ki vključuje občutja, izključuje pa čustva.

krmarjenje (*steering*) Najvišja stopnja vizualizacije podatkov za numerične simulacije, kjer se simulacija vodi s spreminjanjem simulacijskih parametrov med izvajanjem računalniških obdelav.

MIDI sistem Sistem, ki pretvori računalnik v napravo za snemanje, editiranje in predvajanje zvoka.

multimedia Je pojem, kjer se sreča več medijev kot so televizija, film, audio tehnika, založništvo in računalnik.

naknadno procesiranje (*postprocessing*) Prva stopnja vizualizacije podatkov za numerične simulacije, kjer se podatki predstavijo po končanih numeričnih simulacijah.

programiranje vizualizacije (*Program Visualization*) Animacija algoritmov, ki uporablja slike za predstavitev določenih vidikov izvajanj programov.

sintaksno usmerjeno editiranje Editor pozna sintakso jezika in obvesti uporabnika o sintaksni napaki ali pa ga prisili, da uporablja strukturo, ki temelji na sintaksi jezika.

sledenje (*tracking*) Druga stopnja vizualizacije podatkov za numerične simulacije, kjer se prikazujejo rezultati računalniških obdelav v realnem času.

specifikacijsko usmerjeno editiranje Strukturno osnovani editor se razširi s pravili o strukturi programa, za katerega mora obstajati specifikacija.

steropsis Zlitje različnih podob (levega in desnega očesa) tako, da vidimo svet v globino.

vizualizacija Predstavitev podatkov, modelov, konstrukcij in besedil s pomočjo slike.

vizualizacija v računalništvu Predstavitev rezultatov in informacij s pomočjo slike in animacije slik ter vizualen način dela z računalnikom.

vizualizacija v znanosti (*ViSC*) Vizualna predstavitev podatkov, ki so rezultat superračunalniških simulacij, satelitov, in merilnih naprav uporabljenih v astronomiji, meteorologiji, geologiji, medicini in podobno.

vizualizacijska funkcija $\mathcal{V}$ Funkcija, ki iz nekega računskega stanja vodi k sliki.

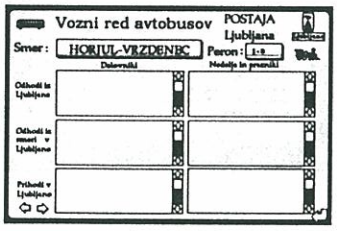
vizualno programiranje (*Visual Programming*) Jeziki, s pomočjo katerih je mogoče programirati v notaciji, ki uporablja dve ali tri dimenzije.

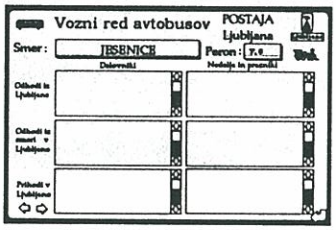
XCMD *(external command)* Procedura, izdelana v programskem jeziku *Pascal* ali *C*, ki jo je mogoče klicati iz programskega okolja *HyperCard* s programskim jezikom *HyperTalk*.

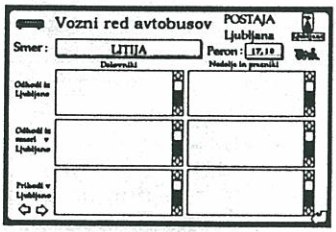
XFNC *(external function)* Funkcija, izdelana v programskem jeziku *Pascal* ali *C*, ki jo je mogoče klicati iz programskega okolja *HyperCard* s programskim jezikom *HyperTalk*.

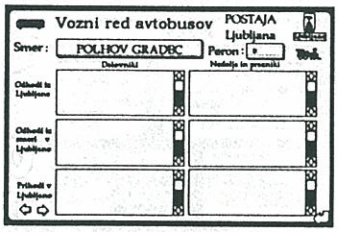
B. Dodatek

Kartice v aplikaciji Ljubljana







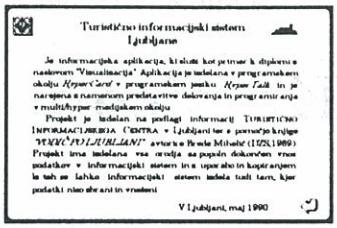



Trg osvobodtve, Zvezda

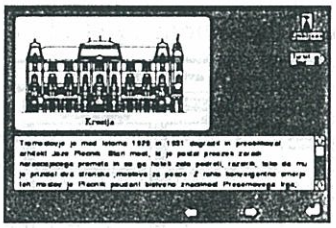

HOTELI

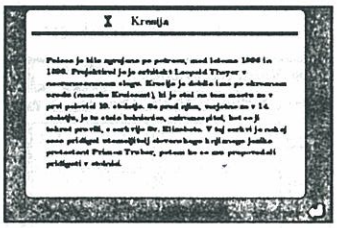

GOSTILNE

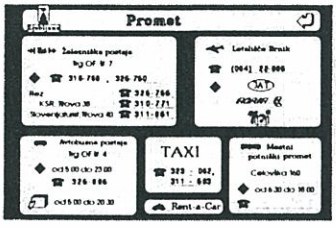

RESTAVRACIJE


Turistično informacijski sistem Ljubljane


Zahvala

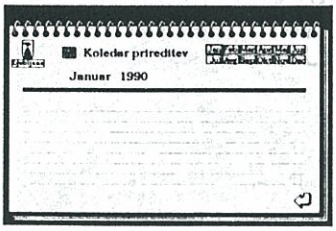


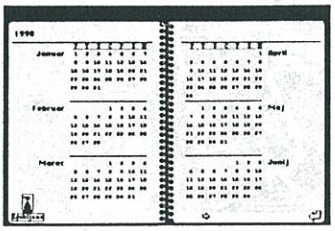

Kresija

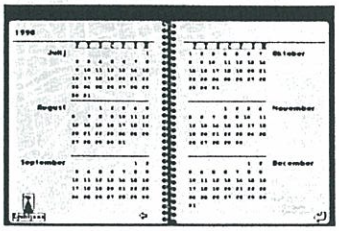

Promet
TAXI

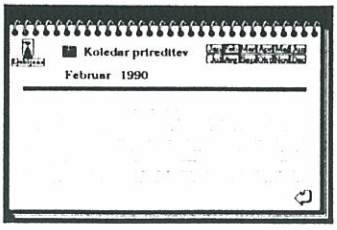

ZOO
Živalski vrt
Večna pot 70
☎ 266-766
Odprto: poleti 7-19
pozimi 8-16

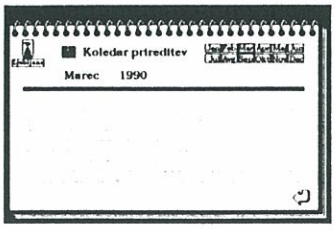

TURISTIČNO INFORMACIJSKI CENTER
Titova 11, 61000 Ljubljana ☎ 215 417, 224 277

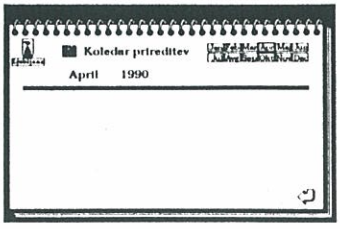

Koledar prireditev
Januar 1990

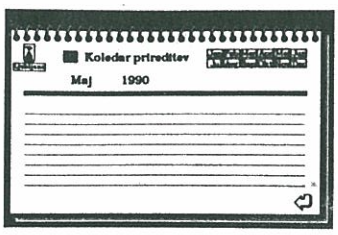


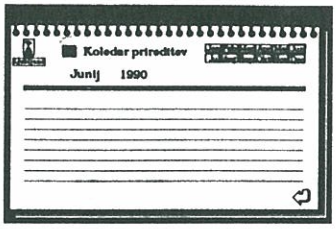


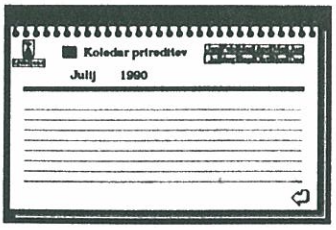

Koledar prireditev
Februar 1990

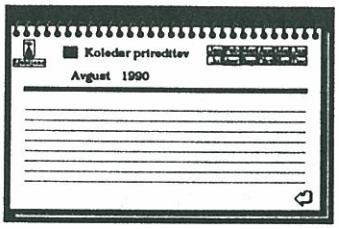

Koledar prireditev
Marec 1990

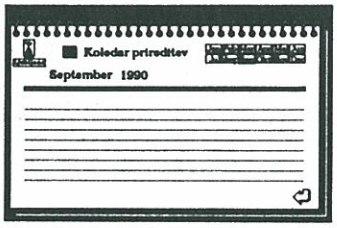

Koledar prireditev
April 1990


Koledar prireditev
Maj 1990

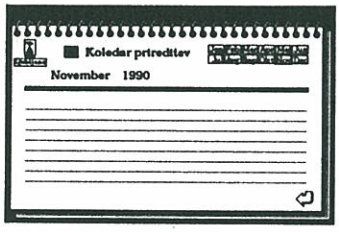

Koledar prireditev
Junij 1990

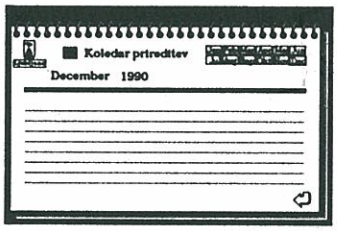

Koledar prireditev
Julij 1990

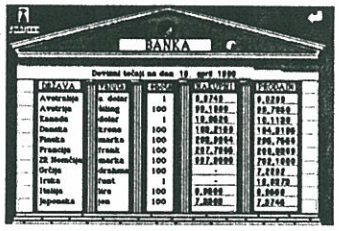

Koledar prireditev
Avgust 1990

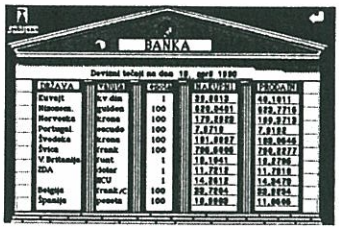

Koledar prireditev
September 1990


Koledar prireditev
Oktober 1990

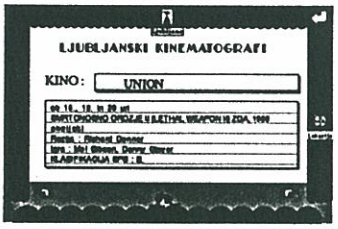

Koledar prireditev
November 1990


Koledar prireditev
December 1990


BANKA


BANKA

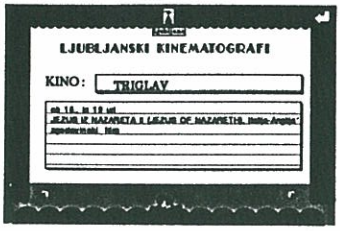

LJUBLJANSKI KINEMATOGRAFI
KINO: KINOTEKA


LJUBLJANSKI KINEMATOGRAFI
KINO: UNION

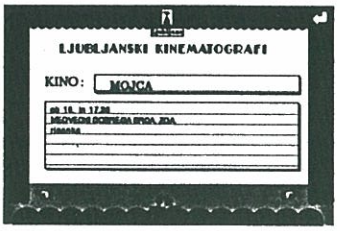

LJUBLJANSKI KINEMATOGRAFI
KINO: MINI KINO UNION


LJUBLJANSKI KINEMATOGRAFI
KINO: KOMUNA


LJUBLJANSKI KINEMATOGRAFI
KINO: BEZIGRAD


LJUBLJANSKI KINEMATOGRAFI
KINO: TRIGLAV


LJUBLJANSKI KINEMATOGRAFI
KINO: VIC


LJUBLJANSKI KINEMATOGRAFI
KINO: MOJCA


LJUBLJANSKI KINEMATOGRAFI
KINO:
BLOGA


Stari del mesta med reko in gradom
Konec


Stari del mesta med reko in gradom
Konec


Stari del mesta med reko in gradom
Konec


Stari del mesta med reko in gradom
Konec

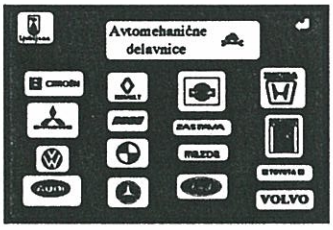

Stari del mesta med reko in gradom
Konec

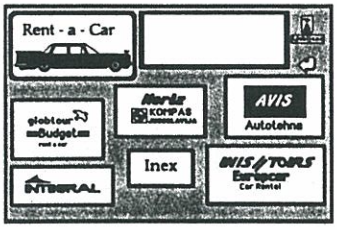

Stari del mesta med reko in gradom
Konec


Stari del mesta med reko in gradom
Konec


Stari del mesta med reko in gradom
Konec


Turistični sprehodi po
Ljubljani
Stari del mesta med reko in gradom
Administrativni in kulturni del med Tivolo cesto in parkom Tivoli
Trg osvoboditve in stari del mesta na levem bregu Ljubljanice
Severni del Ljubljane med Tivolo in Miklošičevo cesto
Mirje, Trnovo, Krakovo
Park Tivoli in hrib Rožnik
Šempeter, Tabor, Potjane
Beograd

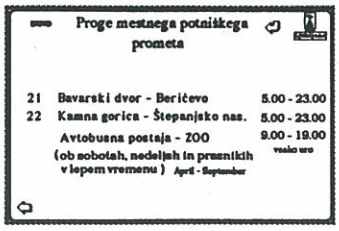

Avtomehanične delavnice
CITROEN
FIAT
ZASTAVA
RENAULT
VOLVO


Rent - a - Car
globtour budget
Merkur
KOMPAS
AVIS
Autolahna
Inex
INTERRAL
WILSHIRE
Car Rental


Proge mestnega potniškega prometa
1 Vižmarje - Mestni log 5.00 - 23.00
2 Nove Jarše - Letališka 5.00 - 23.00
3 Litostroj - Rudnik 5.00 - 21.00
3a Bavarski dvor - Rudnik 21.00 - 23.00
5 Podutik - Kodeljevo 5.00 - 21.00
5a Bavarski dvor - Podutik 21.00 - 23.00
6 Črnuče - Dolgi most 5.00 - 23.00


Proge mestnega potniškega prometa
7 Pržanj - Nove Jarše 5.00 - 23.00
8 Brod - Ježica 5.00 - 21.00
8a Bavarski dvor - Brod 21.00 - 23.00
9 Trnovo - Štepanjsko naselje 5.00 - 23.00
10 Trg osvoboditve - Zadobrova 5.00 - 23.00
11 Bežigrad - Zalog 5.00 - 23.00
12 Bežigrad - Vevče 5.00 - 23.00


Proge mestnega potniškega prometa
13 Trg osvoboditve - Sostro 5.00 - 23.00
14 Savlje - Vrhovci 5.00 - 23.00
15 Bavarski dvor - Medvode 5.00 - 23.00
16 Bavarski dvor - Gameljne 5.00 - 23.00
18 Kino Šiška - Tovarna Lek 5.00 - 16.00
19 Tomačevo - Barje 5.00 - 23.00
20 Stožice BS3 - Fužine 5.00 - 23.00


Proge mestnega potniškega prometa
21 Bavarski dvor - Beričevo 5.00 - 23.00
22 Kamna gorica - Štepanjsko nas. 5.00 - 23.00
Avtobusna postaja - ZOO 9.00 - 19.00
(ob sobotah, nedeljah in prazničnih v lepem vremenu) April - September
vsako uro


Turistične agencije
ATLAS
ATLAS
Mestni trg 8
222 - 711

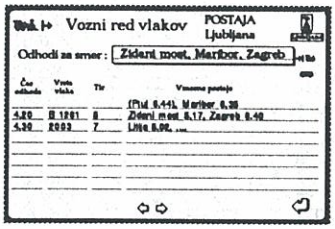

Turistične agencije
GENERALTURIST
Gosposvetska 7
217 - 223

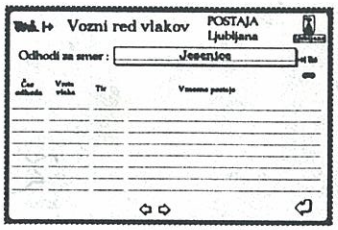

Turistične agencije
Globtour
globtour
Trg revolucije 1 (Mestni trg)
Gosposvetska 4
Rent-a-Car
213 - 912


Turistične agencije
KOMPAS
KOMPAS
Miklošičeva 11
Tržaška 12
Rent-a-Car
326 - 463


Vozni red vlakov
POSTAJA Ljubljana
Odhodi za smer:
Zidani most, Maribor, Zagreb
VENEZIJA EXPRES ; Zagreb gl. kol. 3.10.
Slavonski kred 5.06., Vinkovci 5.40.,
Beograd 9.05, Niš 13.14, Skopje 16.43,
Gevgelija 18.30, Solun(Thessaloniki)
23.10, Atene(Athenai) 8.00
ISTANBUL EKSPRES: Zagreb gl. kol. 3.23,
Slavonski Brod 5.46, Vinkovci 6.45,
Beograd 9.40, Niš 12.21, Nadalinjenje :>

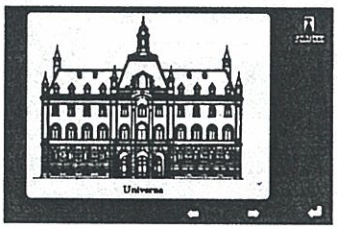

Vozni red vlakov
POSTAJA Ljubljana
Odhodi za smer:
Zidani most, Maribor, Zagreb
Dimitrovgrad 16.16, Sofija 19.26,
Svilengrad 1.12, Istanbul 10.10
3.40 BP 101 3 SAVA EKSPRES: Zidani most 3.34, Zagreb
4.01, Slavonski Brod 6.01, Vinkovci 7.41,
Beograd 9.30
3.40 2102 8 Posebni vlak do Zagreba, Litija 4.10,
Zidani most 4.46, (Dobova 5.40, Zagreb
6.10),Celje 8.20, Pragersko 9.10 , Nadalj.

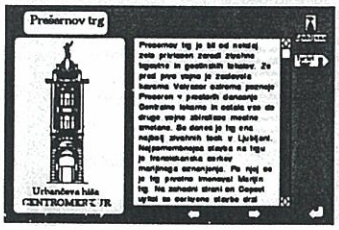

Vozni red vlakov
POSTAJA Ljubljana
Odhodi za smer:
Zidani most, Maribor, Zagreb
(Piši 5.44), Maribor 6.35
4.20 B 1201 8 Zidani most 6.17, Zagreb 8.40
4.30 2003 7 Litija 5.02, ...

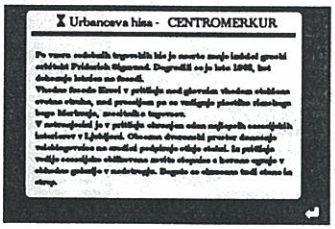

Vozni red vlakov
POSTAJA Ljubljana
Odhodi za smer:
Jesenice

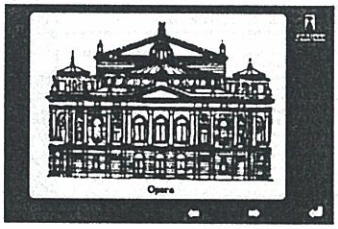

Vozni red vlakov
POSTAJA Ljubljana
Prihodi iz smeri :
Zidani most, Maribor, Zagreb

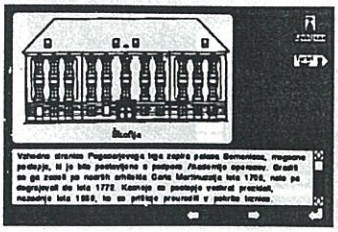

Klic telefonske številke
323 - 082

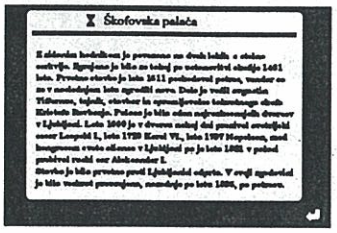

Univerza

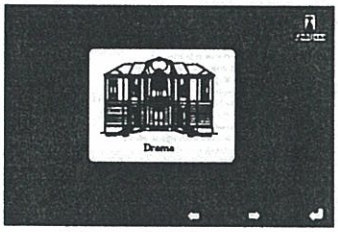

Prešernov trg
Urbančeva hiša
CENTROMERKUR


Urbančeva hisa - CENTROMERKUR


Opera

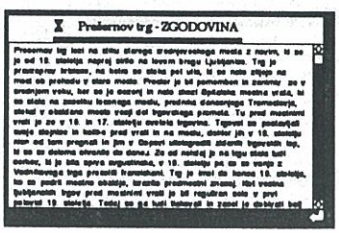

Škofja

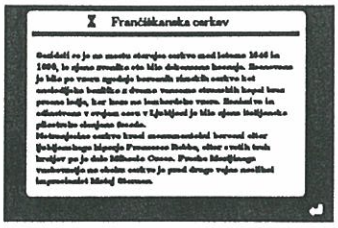

Škofovska palača

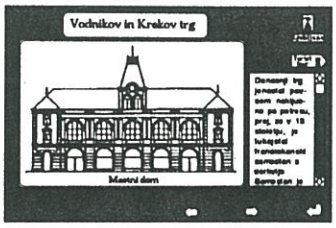

Drama

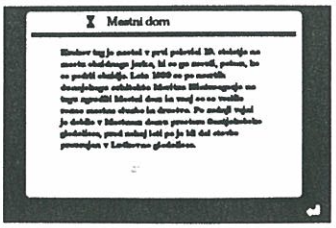

Slovenska filharmonija

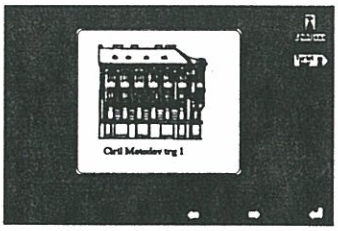

Prešernov trg
Frančiškanska cerkev marijinega oznanjenja


Prešernov trg - ZGODOVINA

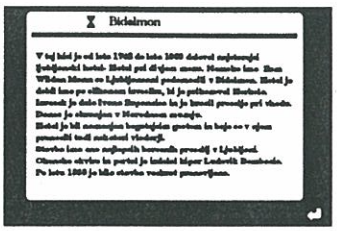

Frančiškanska cerkev

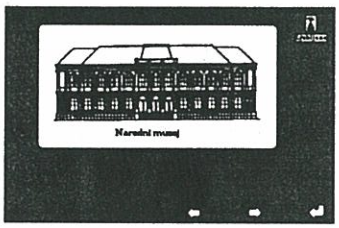

Vodnikov in Krekov trg
Mestni dom

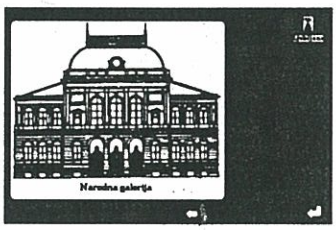

Mestni dom

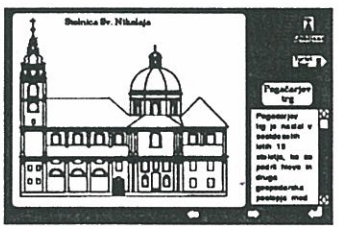

Ciril Metodov trg 1

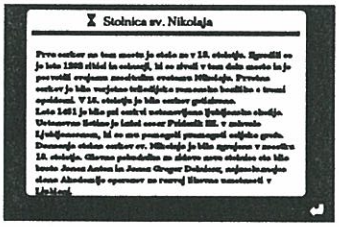

Ciril-Metodov trg
Ciril-Metodov trg je del
Bidermon

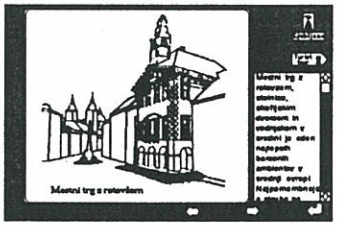

Bidermon

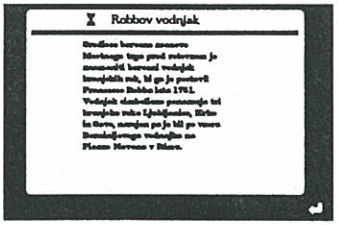

Narodni muzej

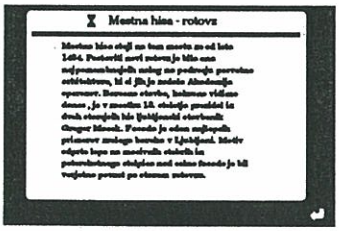

Narodna galerija

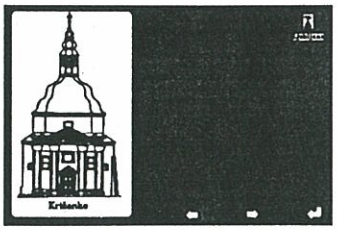

Stolnica Sv. Nikolaja
Pogačarjev trg


Stolnica sv. Nikolaja

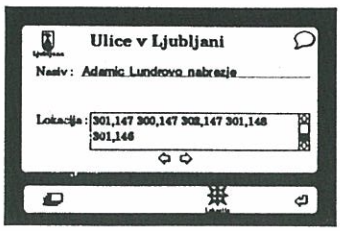

Mestni trg z rotovžem


Robbov vodnjak

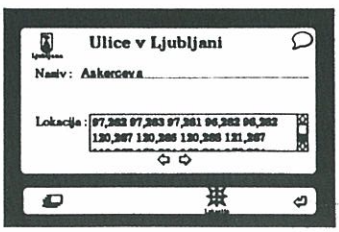

Mestna hiša - rotovž


Križanke

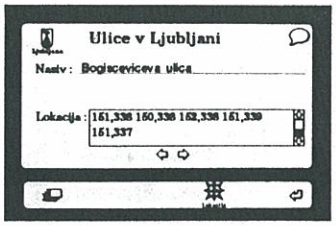

Uršulinska cerkev

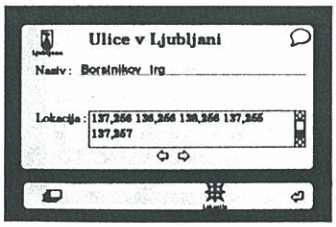

Ulice v Ljubljani
Naziv : Adamic Lundrovo nabrežje
Lokacija : 301,147 300,147 302,147 301,148 301,146

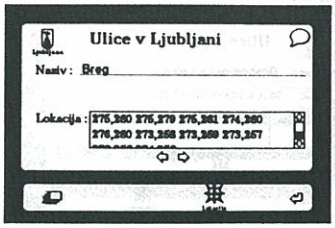

Ulice v Ljubljani
Naziv : Ambrozicev trg
Lokacija : 465,162 466,162 464,162 165,161 466,163

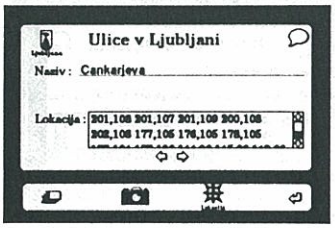

Ulice v Ljubljani
Naziv : Askerceva
Lokacija : 97,283 97,283 97,281 96,282 96,282 120,287 120,286 120,286 121,287

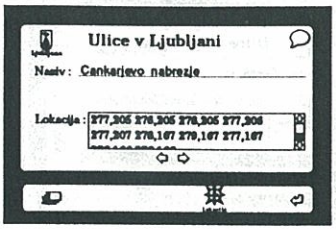

Ulice v Ljubljani
Naziv: Beethovnova
Lokacija: 183,142 183,142 184,142 183,141
183,143 187,118 188,118 188,118

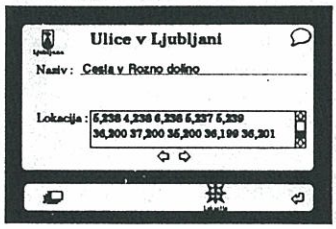

Ulice v Ljubljani
Naziv: Bogiscevicева ulica
Lokacija: 151,338 150,338 152,338 151,339
151,337

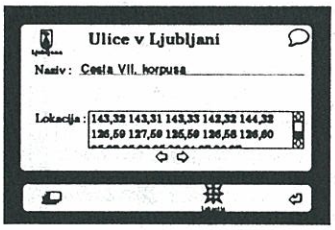

Ulice v Ljubljani
Naziv: Borstnikov trg
Lokacija: 137,256 136,256 138,256 137,255
137,257


Ulice v Ljubljani
Naziv: Breg
Lokacija: 276,280 276,279 276,281 274,280
276,280 273,268 273,269 273,267


Ulice v Ljubljani
Naziv: Cankarjeva
Lokacija: 201,108 201,107 201,109 200,108
202,108 177,105 176,105 178,105

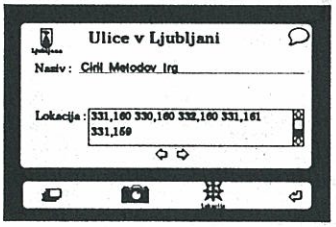

Ulice v Ljubljani
Naziv: Cankarjevo nabrezje
Lokacija: 277,205 276,205 278,205 277,206
277,207 276,167 279,167 277,167

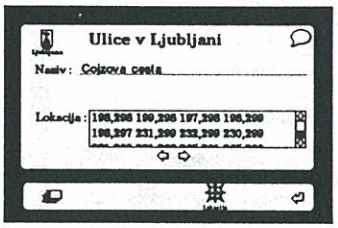

Ulice v Ljubljani
Naziv: Cesta v Rozno dolino
Lokacija: 5,238 4,238 6,238 5,237 5,239
36,200 37,200 35,200 36,199 36,201

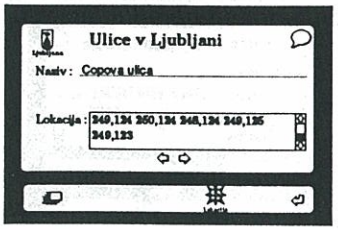

Ulice v Ljubljani
Naziv: Cesta VII. korpusa
Lokacija: 143,32 143,31 143,33 142,32 144,32
126,59 127,59 125,59 126,58 126,60

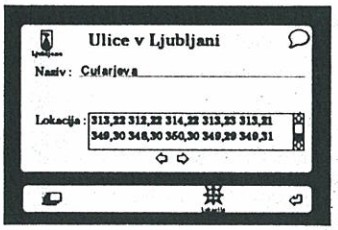

Ulice v Ljubljani
Naziv: Cevljarska
Lokacija: 258,238 259,238 257,238 258,237
258,239

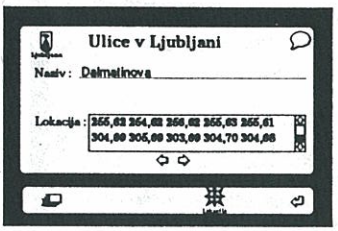

Ulice v Ljubljani
Naziv: Cigaletova
Lokacija: 266,34 267,34 266,34 266,33 266,35

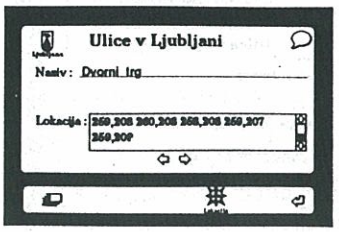

Ulice v Ljubljani
Naziv: Ciril Metodov trg
Lokacija: 331,160 330,160 332,160 331,161
331,159

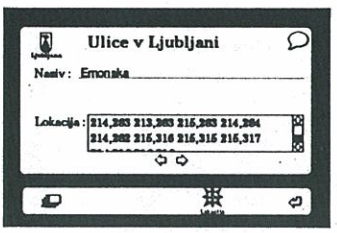

Ulice v Ljubljani
Naziv: Cojzova cesta
Lokacija: 196,296 196,296 197,296 196,299
196,297 231,299 232,299 230,299

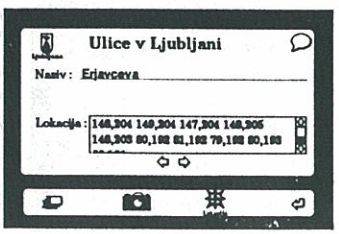

Ulice v Ljubljani
Naziv: Copova ulica
Lokacija: 249,124 250,124 248,124 249,125
249,123


Ulice v Ljubljani
Naziv: Cufarjeva
Lokacija: 313,22 312,22 314,22 313,23 313,21
349,30 348,30 350,30 349,29 349,31

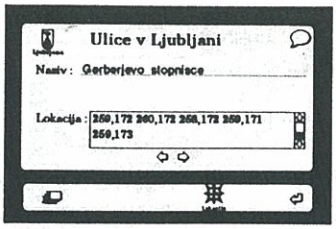

Ulice v Ljubljani
Naziv: Dalmatinova
Lokacija: 255,62 254,62 256,62 255,63 255,61
304,69 305,69 303,69 304,70 304,68

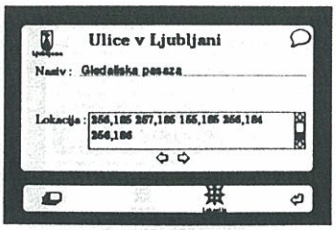

Ulice v Ljubljani
Naziv: Dvorni trg
Lokacija: 259,208 260,208 258,208 259,207
259,209

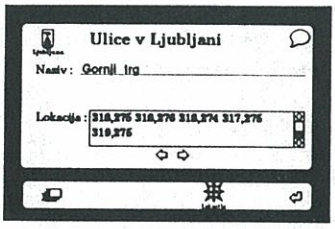

Ulice v Ljubljani
Naziv: Emonska
Lokacija: 214,283 213,283 215,283 214,284
214,282 215,316 215,315 215,317

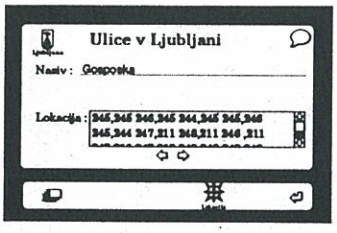

Ulice v Ljubljani
Naziv: Erjavceva
Lokacija: 148,304 149,304 147,304 148,305
148,303 80,192 81,192 79,192 80,193

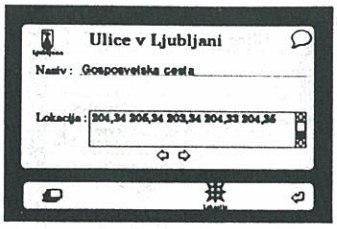

Ulice v Ljubljani
Naziv: Gallusovo nabrezje
Lokacija: 291,279 292,279 290,279 291,278
291,280 289,354 290,354 288,354

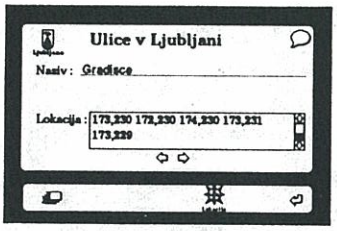

Ulice v Ljubljani
Naziv: Gerberjevo stopnisce
Lokacija: 259,172 260,172 258,172 259,171
259,173

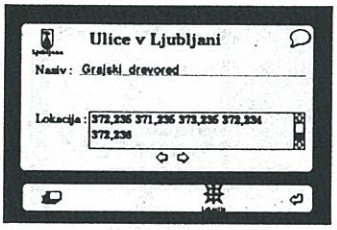

Ulice v Ljubljani
Naziv: Gledaliska pesaza
Lokacija: 356,185 357,185 355,185 356,184
356,186

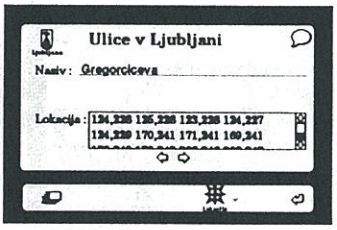

Ulice v Ljubljani
Naziv: Gornji trg
Lokacija: 318,276 318,276 318,274 317,276
319,276

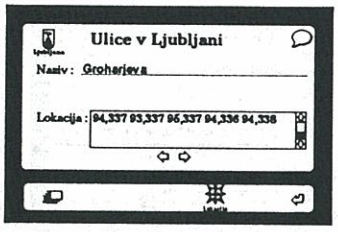

Ulice v Ljubljani
Naziv: Gosposka
Lokacija: 345,345 346,345 344,345 345,346
345,344 347,211 348,211 346,211

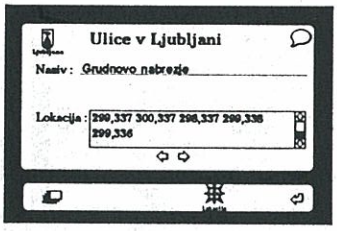

Ulice v Ljubljani
Naziv: Gosposvetska cesta
Lokacija: 204,34 206,34 203,34 204,33 204,35

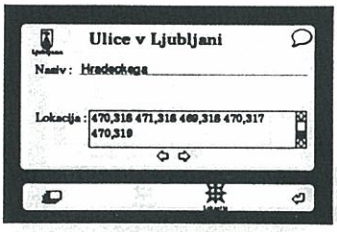

Ulice v Ljubljani
Naziv: Gradisce
Lokacija: 173,230 172,230 174,230 173,231
173,229

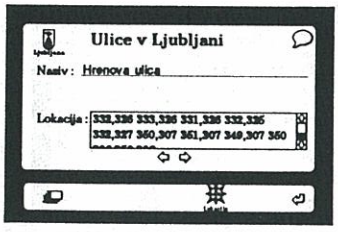

Ulice v Ljubljani
Naziv: Grajski drevored
Lokacija: 372,235 371,236 373,236 372,234
372,236

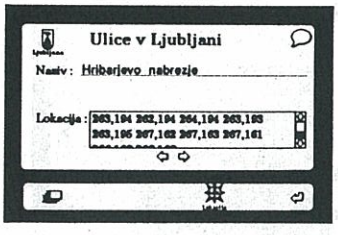

Ulice v Ljubljani
Naziv: Gregorciceva
Lokacija: 124,238 125,238 123,238 124,237
124,239 170,341 171,341 169,341

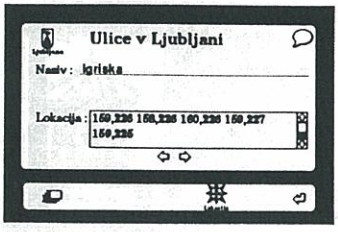

Ulice v Ljubljani
Naziv: Groharjeva
Lokacija: 94,337 93,337 95,337 94,336 94,338

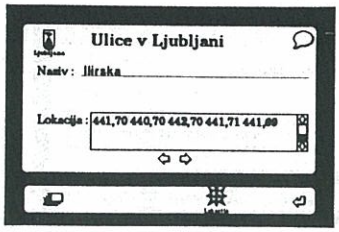

Ulice v Ljubljani
Naziv: Grudnovo nabrezje
Lokacija: 299,337 300,337 298,337 299,336
299,336

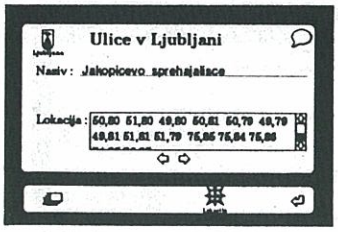

Ulice v Ljubljani
Naziv: Hradeckega
Lokacija: 470,318 471,318 469,318 470,317
470,319

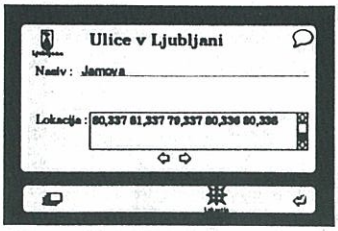

Ulice v Ljubljani
Naziv: Hrenova ulica
Lokacija: 332,336 333,336 331,336 332,335
332,337 360,307 361,307 349,307 350

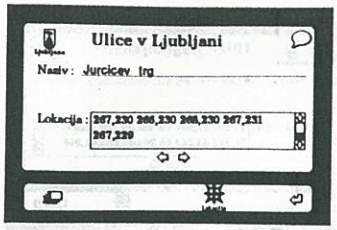

Ulice v Ljubljani
Naziv: Hriberjevo nabrezje
Lokacija: 263,194 262,194 264,194 263,193
263,195 267,162 267,163 267,161

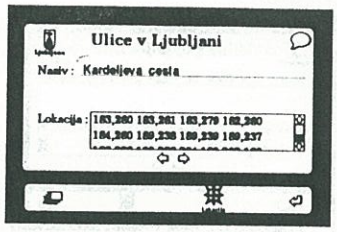

Ulice v Ljubljani
Naziv: Igriska
Lokacija: 159,236 158,236 160,236 159,237
159,235


Ulice v Ljubljani
Naziv: Ilirska
Lokacija: 441,70 440,70 442,70 441,71 441,69

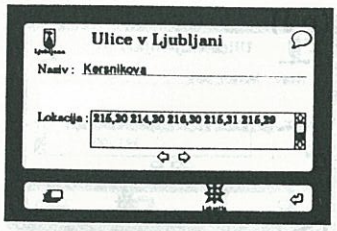

Ulice v Ljubljani
Naziv: Jakopicevo sprehajalisce
Lokacija: 50,80 51,80 49,80 50,81 50,79 49,79
49,81 51,81 51,79 75,85 75,84 75,86

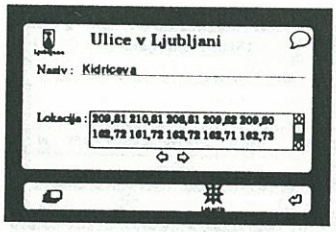

Ulice v Ljubljani
Naziv: Jamova
Lokacija: 80,337 81,337 79,337 80,336 80,338

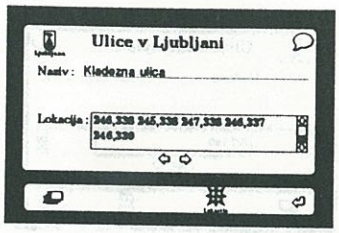

Ulice v Ljubljani
Naziv: Jurcicev trg
Lokacija: 267,230 266,230 268,230 267,231 267,229

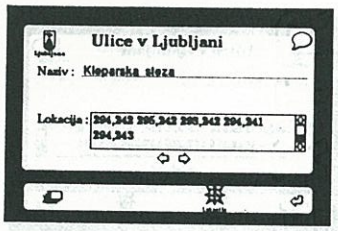

Ulice v Ljubljani
Naziv: Kardeljeva cesta
Lokacija: 183,280 183,281 183,279 182,280 184,280 189,238 189,239 189,237

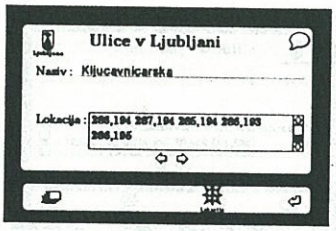

Ulice v Ljubljani
Naziv: Karlovska cesta
Lokacija: 313,296 313,299 313,297 312,298 314,298 337,309 336,309 336,309

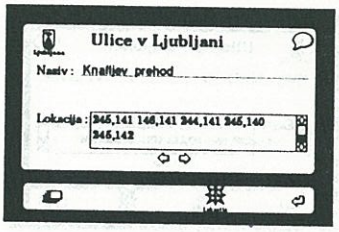

Ulice v Ljubljani
Naziv: Kersnikova
Lokacija: 215,30 214,30 216,30 215,31 215,29

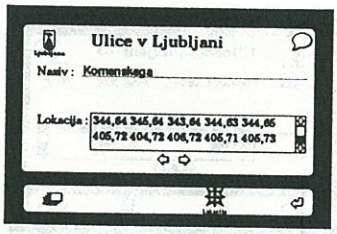

Ulice v Ljubljani
Naziv: Kidriceva
Lokacija: 209,81 210,81 208,81 209,82 209,80 162,72 161,72 163,72 162,71 162,73

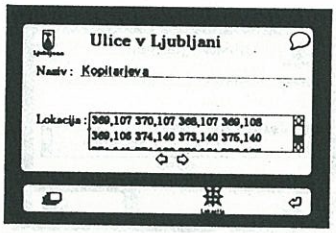

Ulice v Ljubljani
Naziv: Kladezna ulica
Lokacija: 346,338 345,336 347,338 346,337 346,339

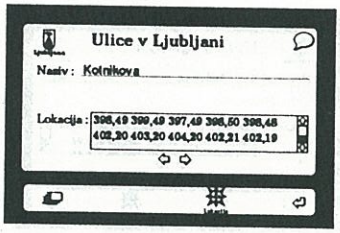

Ulice v Ljubljani
Naziv: Kleparska steza
Lokacija: 294,343 295,343 293,343 294,341 294,343

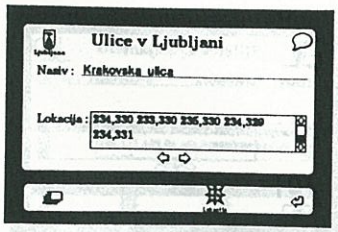

Ulice v Ljubljani
Naziv: Kljucavnicarska
Lokacija: 286,194 287,194 285,194 286,193 286,195

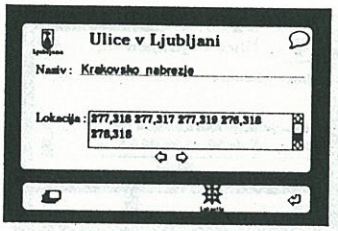

Ulice v Ljubljani
Naziv: Knafljev prehod
Lokacija: 345,141 146,141 344,141 345,140 345,142

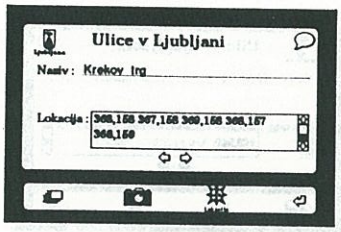

Ulice v Ljubljani
Naziv: Komenskega
Lokacija: 344,64 345,64 343,64 344,63 344,65 405,72 404,72 406,72 405,71 405,73

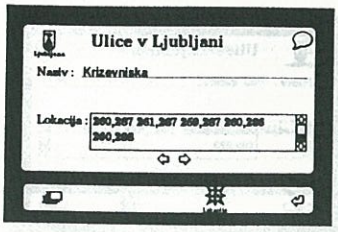

Ulice v Ljubljani
Naziv: Kopitarjeva
Lokacija: 369,107 370,107 368,107 369,108 369,108 374,140 373,140 375,140

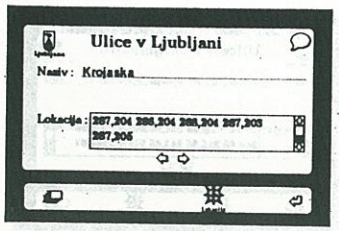

Ulice v Ljubljani
Naziv: Kotnikova
Lokacija: 398,49 399,49 397,49 398,50 398,48 402,20 403,20 404,20 402,21 402,19

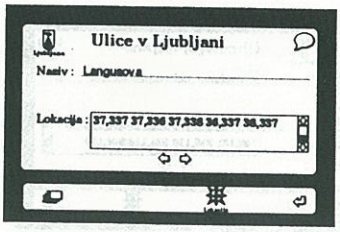

Ulice v Ljubljani
Naziv: Krakovska ulica
Lokacija: 234,330 233,330 235,330 234,329 234,331

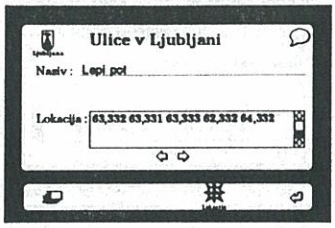

Ulice v Ljubljani
Naziv: Krekovsko nabrezje
Lokacija: 277,318 277,317 277,319 276,318 276,318

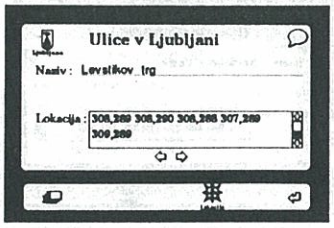

Ulice v Ljubljani
Naziv: Krekov trg
Lokacija: 368,168 367,168 369,168 368,167 368,169

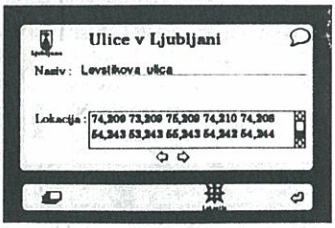

Ulice v Ljubljani
Naziv: Krizevniska
Lokacija: 260,267 261,267 259,267 260,266 260,268


Ulice v Ljubljani
Naziv: Krojaska
Lokacija: 287,204 286,204 288,204 287,203 287,205


Ulice v Ljubljani
Naziv: Langusova
Lokacija: 37,337 37,336 37,338 36,337 38,337


Ulice v Ljubljani
Naziv: Lepi pot
Lokacija: 63,332 63,331 63,333 62,332 64,332

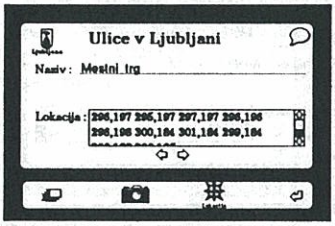

Ulice v Ljubljani
Naziv: Levstikov trg
Lokacija: 308,289 308,290 308,288 307,289 309,289

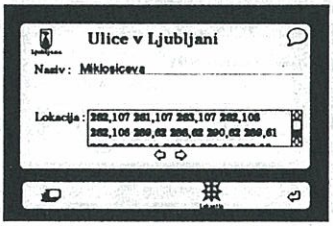

Ulice v Ljubljani
Naziv: Levstikova ulica
Lokacija: 74,309 73,309 75,309 74,310 74,308 54,343 53,343 55,343 54,342 54,344

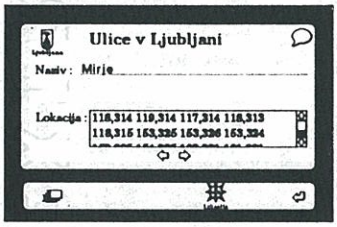

Ulice v Ljubljani
Naziv: Mackova
Lokacija: 307,161 308,161 308,161 307,162 307,160

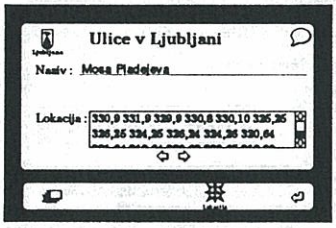

Ulice v Ljubljani
Naziv: Maistrova
Lokacija: 460,32 461,32 459,32 460,31 460,33 460,55 461,55 459,55 460,54 460,56

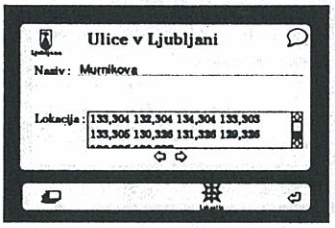

Ulice v Ljubljani
Naziv: Marxov park
Lokacija: 276,56 277,56 275,56 276,55 276,57

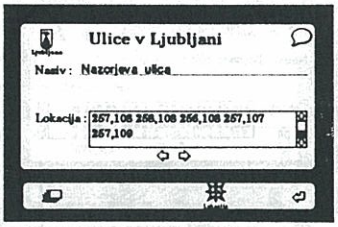

Ulice v Ljubljani
Naziv: Mestni trg
Lokacija: 296,197 295,197 297,197 296,196 298,196 300,184 301,184 299,184

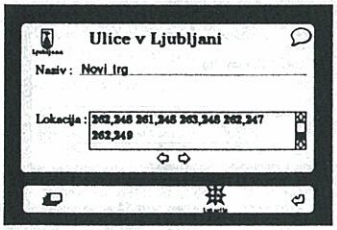

Ulice v Ljubljani
Naziv: Miklosiceva
Lokacija: 282,107 281,107 283,107 282,108 282,106

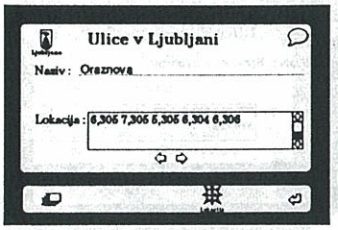

Ulice v Ljubljani
Naziv: Mirje
Lokacija: 118,314 119,314 117,314 118,313 118,315 153,335 153,336 153,334

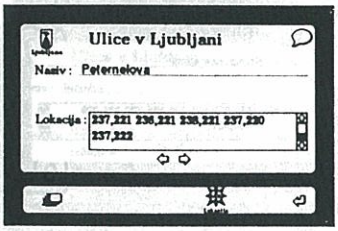

Ulice v Ljubljani
Naziv: Mosa Piadejeva
Lokacija: 330,9 331,9 329,9 330,8 330,10 325,25 326,25 324,25 326,24 324,26 320,64

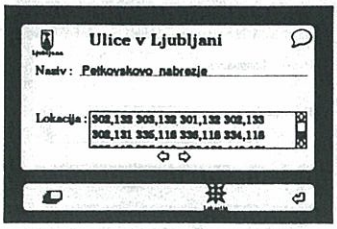

Ulice v Ljubljani
Naziv: Murnikova
Lokacija: 133,304 132,304 134,304 133,303 133,305 130,336 131,336 129,336

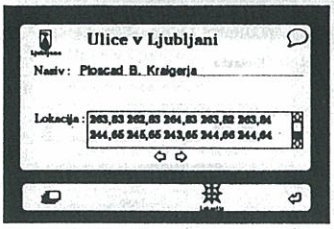

Ulice v Ljubljani
Naziv: Nazorjeva ulica
Lokacija: 257,108 258,108 256,108 257,107 257,109

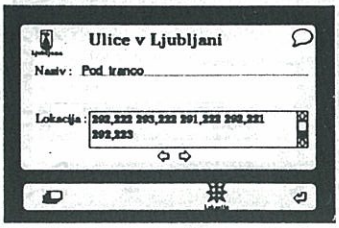

Ulice v Ljubljani
Naziv: Novi trg
Lokacija: 262,348 261,348 263,348 262,347 262,349


Ulice v Ljubljani
Naziv: Oraznova
Lokacija: 6,305 7,305 5,305 6,304 6,306

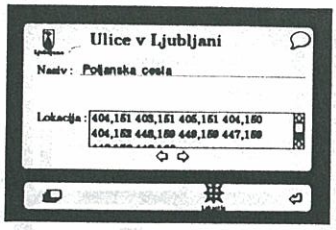

Ulice v Ljubljani
Naziv: Petemelova
Lokacija: 237,221 236,221 238,221 237,220 237,222

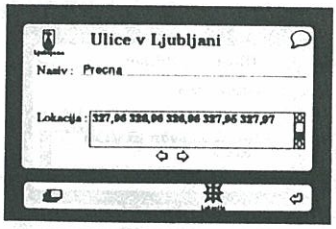

Ulice v Ljubljani
Naziv: Petkovskovo nabrezje
Lokacija: 308,133 308,132 307,132 308,133 308,131 336,116 336,116 334,116

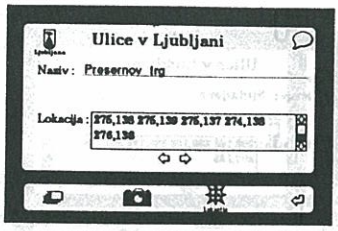

Ulice v Ljubljani
Naziv: Ploscad B. Kraigerja
Lokacija: 343,83 342,83 344,83 343,82 343,84 344,65 345,65 343,65 344,66 344,64

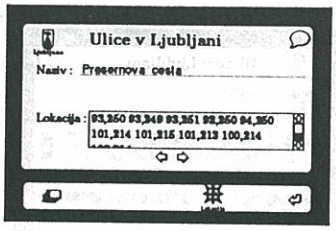

Ulice v Ljubljani
Naziv: Pod tranco
Lokacija: 292,222 293,222 291,222 292,221 292,223

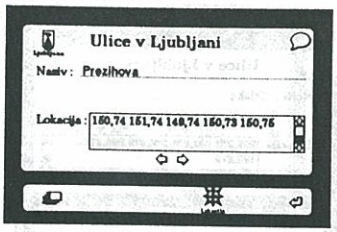

Ulice v Ljubljani
Naziv: Pogacarjev trg
Lokacija: 317,144 318,144 316,144 317,145 317,143

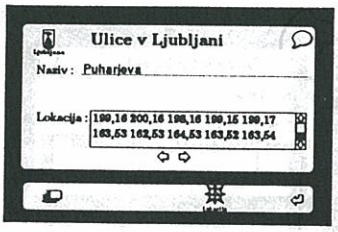

Ulice v Ljubljani
Naziv: Poljanska cesta
Lokacija: 404,151 405,151 405,151 404,150 404,152 448,189 449,189 447,189

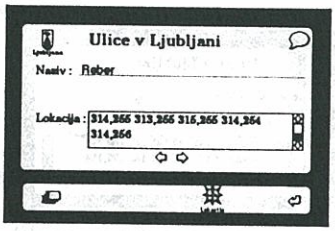

Ulice v Ljubljani
Naziv: Precna
Lokacija: 327,96 328,96 326,96 327,95 327,97

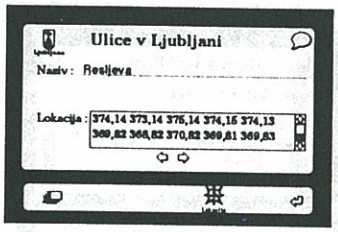

Ulice v Ljubljani
Naziv: Presernov trg
Lokacija: 276,138 276,139 276,137 274,138 276,138

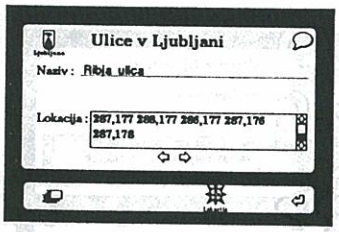

Ulice v Ljubljani
Naziv: Presernova cesta
Lokacija: 93,350 93,349 93,351 93,350 94,350 101,214 101,215 101,213 100,214

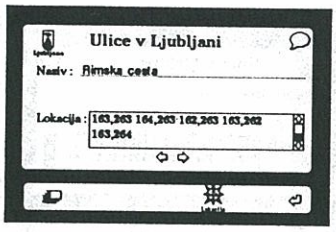

Ulice v Ljubljani
Naziv: Prezihova
Lokacija: 150,74 151,74 149,74 150,73 150,75

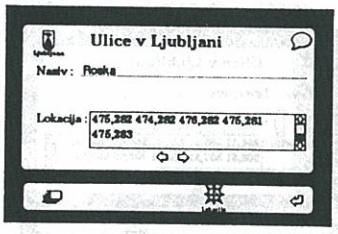

Ulice v Ljubljani
Naziv: Puharjeva
Lokacija: 199,16 200,16 198,16 199,15 199,17 163,53 162,53 164,53 163,52 163,54

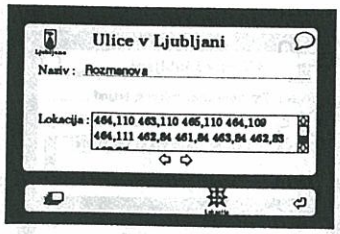

Ulice v Ljubljani
Naziv: Reber
Lokacija: 314,355 313,355 315,355 314,354 314,356


Ulice v Ljubljani
Naziv: Resljeva
Lokacija: 374,14 373,14 376,14 374,15 374,13 369,82 368,82 370,82 369,81 369,83

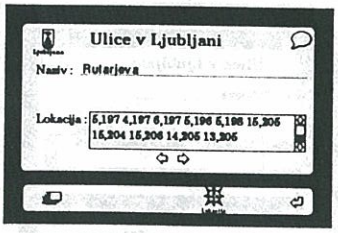

Ulice v Ljubljani
Naziv: Ribja ulica
Lokacija: 287,177 286,177 286,177 287,176 287,178

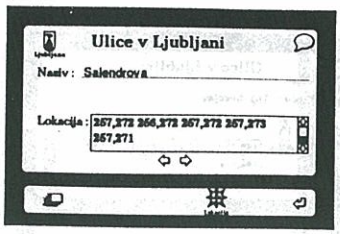

Ulice v Ljubljani
Naziv: Rimska cesta
Lokacija: 163,263 164,263 162,263 163,262 163,264

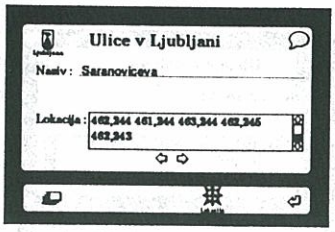

Ulice v Ljubljani
Naziv: Roska
Lokacija: 475,282 474,282 476,282 475,281 475,283

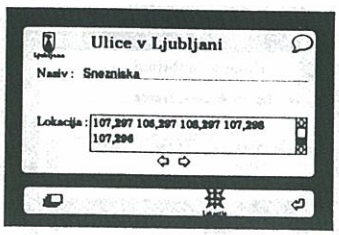

Ulice v Ljubljani
Naziv: Rozmanova
Lokacija: 464,110 463,110 465,110 464,109 464,111 462,84 461,84 463,84 462,83

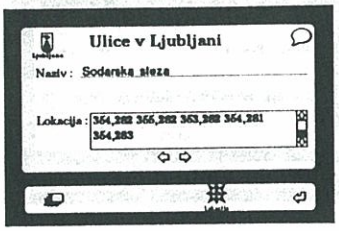

Ulice v Ljubljani
Naziv: Rozna ulica
Lokacija: 318,314 317,314 319,314 318,313 318,315 336,296 336,295 336,297

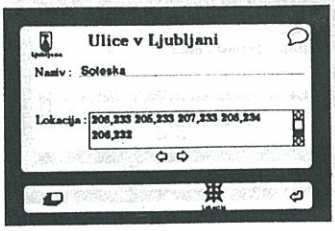

Ulice v Ljubljani
Naziv: Rutarjeva
Lokacija: 5,197 4,197 6,197 5,196 5,198 15,206 15,204 15,206 14,206 13,206

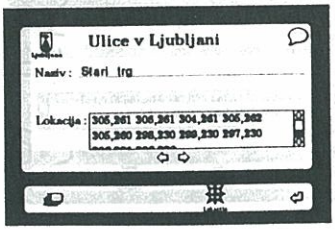

Ulice v Ljubljani
Naziv: Salendrova
Lokacija: 257,272 256,272 257,272 257,273 257,271

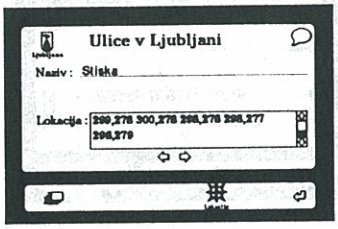

Ulice v Ljubljani
Naziv: Saranovicева
Lokacija: 462,344 461,344 463,344 462,345 462,343

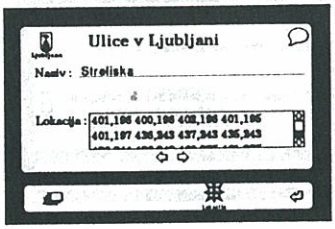

Ulice v Ljubljani
Naziv: Snezniska
Lokacija: 107,297 106,297 108,297 107,298 107,296

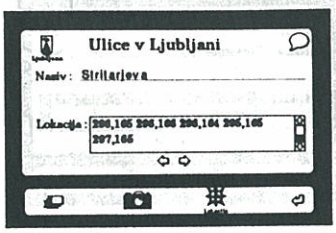

Ulice v Ljubljani
Naziv: Sodarska_steza
Lokacija: 354,282 355,282 353,282 354,281 354,283

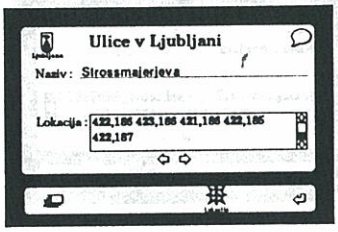

Ulice v Ljubljani
Naziv: Soteska
Lokacija: 206,233 205,233 207,233 206,234 206,232

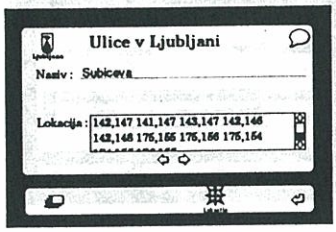

Ulice v Ljubljani
Naziv: Stari_trg
Lokacija: 305,301 306,301 304,301 305,302 305,300 298,230 299,230 297,230

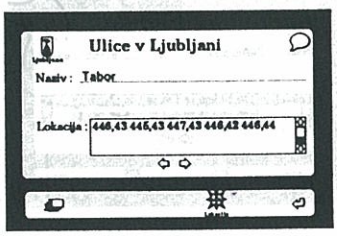

Ulice v Ljubljani
Naziv: Stiska
Lokacija: 299,278 300,278 298,278 298,277 298,279

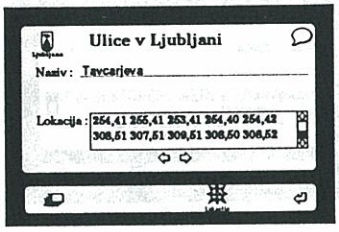

Ulice v Ljubljani
Naziv: Strelisko
Lokacija: 401,196 400,196 402,196 401,195 401,197 436,243 437,243 436,243

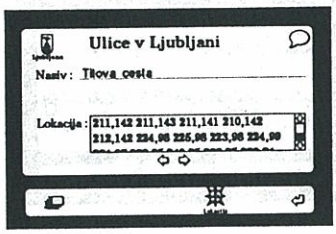

Ulice v Ljubljani
Naziv: Stritarjeva
Lokacija: 298,165 296,166 298,164 296,165 297,165

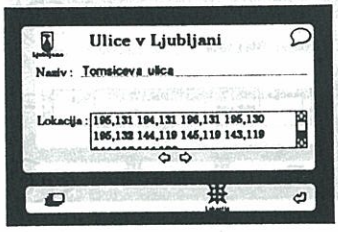

Ulice v Ljubljani
Naziv: Strossmajerjeva
Lokacija: 422,186 423,186 421,186 422,185 422,187

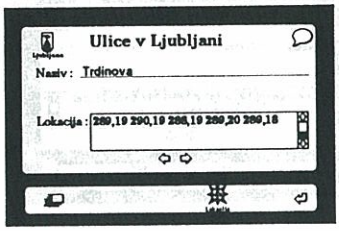

Ulice v Ljubljani
Naziv: Subicova
Lokacija: 142,147 141,147 143,147 142,146 142,148 175,165 175,166 175,164

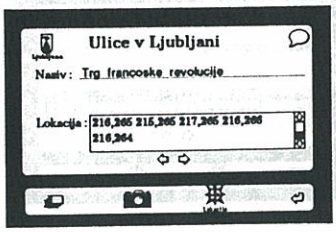

Ulice v Ljubljani
Naziv: Tabor
Lokacija: 446,43 445,43 447,43 446,42 446,44

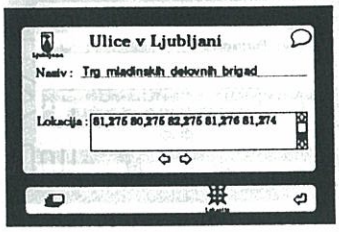

Ulice v Ljubljani
Naziv: Tavcarjeva
Lokacija: 254,41 255,41 253,41 254,40 254,42 308,51 307,51 309,51 308,50 308,52

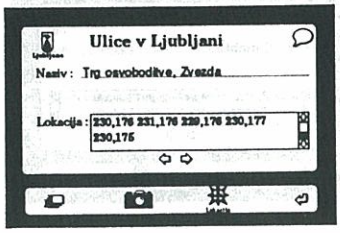

Ulice v Ljubljani
Naziv: Tkova_cesta
Lokacija: 211,142 211,143 211,141 210,142 212,142 224,98 225,98 223,98 224,99

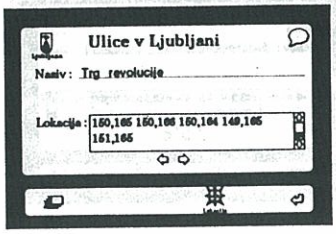

Ulice v Ljubljani
Naziv: Tomsicova_ulica
Lokacija: 195,131 194,131 196,131 195,130 195,132 144,119 145,119 143,119

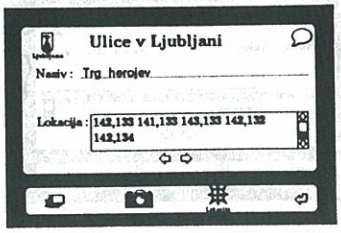

Ulice v Ljubljani
Naziv: Trdinova
Lokacija: 289,19 290,19 288,19 289,20 289,18


Ulice v Ljubljani
Naziv: Trg_francoske_revolucije
Lokacija: 216,265 215,265 217,265 216,266 216,264


Ulice v Ljubljani
Naziv: Trg_mladinskih_delovnih_brigad
Lokacija: 81,275 80,275 82,275 81,276 81,274


Ulice v Ljubljani
Naziv: Trg_osvobodilne_fronte
Lokacija: 230,176 231,176 229,176 230,177 230,175

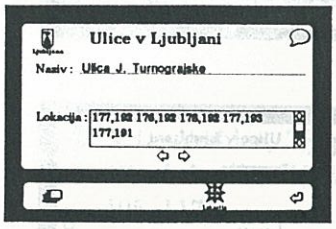

Ulice v Ljubljani
Naziv: Trg_revolucije
Lokacija: 160,165 160,166 160,164 149,165 161,165


Ulice v Ljubljani
Naziv: Trg_herojev
Lokacija: 142,133 141,133 143,133 142,132 142,134

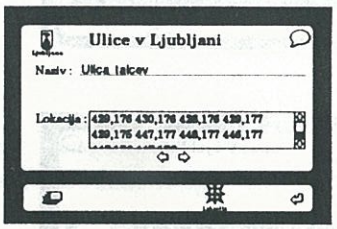

Ulice v Ljubljani
Naziv: Trubarjeva
Lokacija: 389,101 390,101 388,101 389,100
389,102 293,123 294,123 292,123

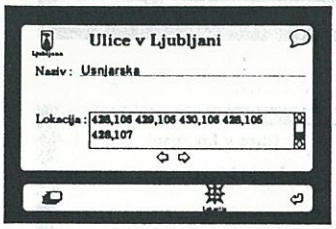

Ulice v Ljubljani
Naziv: Trzaska cesta
Lokacija: 7,333 7,333 7,331 6,333 8,333
36,310 36,310 34,310 36,311 36,309

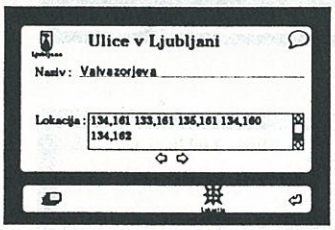

Ulice v Ljubljani
Naziv: Turjaska
Lokacija: 232,351 232,350 232,352 231,351
233,351

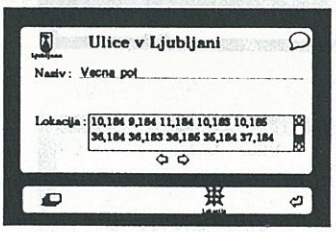

Ulice v Ljubljani
Naziv: Ulica J. Turnograjske
Lokacija: 177,192 176,192 178,192 177,193
177,191

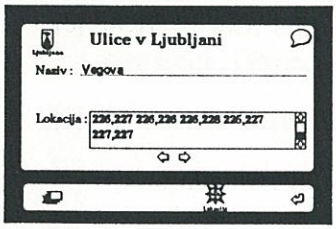

Ulice v Ljubljani
Naziv: Ulica stare pravde
Lokacija: 439,207 439,207 438,207 439,208
439,208 443,212 444,212 442,212

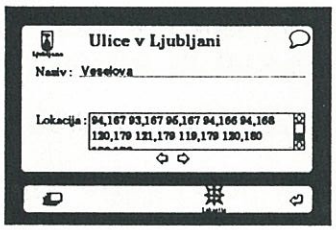

Ulice v Ljubljani
Naziv: Ulica Talcev
Lokacija: 439,176 439,176 438,176 439,177
439,176 447,177 448,177 446,177

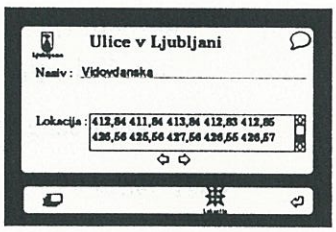

Ulice v Ljubljani
Naziv: Usnjarska
Lokacija: 428,106 429,106 430,106 428,105
428,107


Ulice v Ljubljani
Naziv: Valvazorjeva
Lokacija: 134,161 133,161 135,161 134,160
134,162

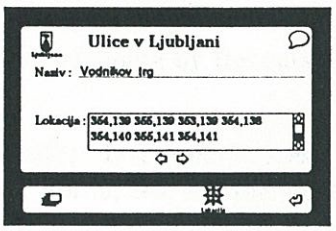

Ulice v Ljubljani
Naziv: Vecna pot
Lokacija: 10,184 9,184 11,184 10,183 10,185
36,184 36,183 36,185 35,184 37,184


Ulice v Ljubljani
Naziv: Vegova
Lokacija: 226,227 226,226 226,228 225,227
227,227

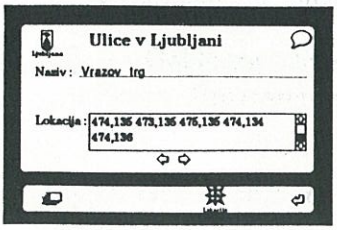

Ulice v Ljubljani
Naziv: Veselova
Lokacija: 94,167 93,167 95,167 94,166 94,168
120,179 121,179 119,179 120,180

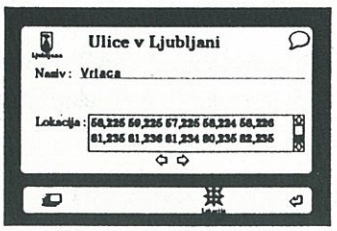

Ulice v Ljubljani
Naziv: Vidovdanska
Lokacija: 412,84 411,84 413,84 412,83 412,85
426,56 425,56 427,56 426,55 426,57

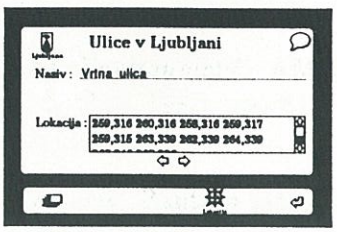

Ulice v Ljubljani
Naziv: Vodna steza
Lokacija: 297,252 298,252 296,252 297,251
297,253

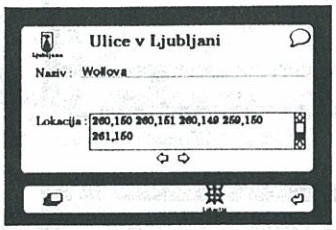

Ulice v Ljubljani
Naziv: Vodnikov trg
Lokacija: 364,139 365,139 363,139 364,138
364,140 365,141 364,141

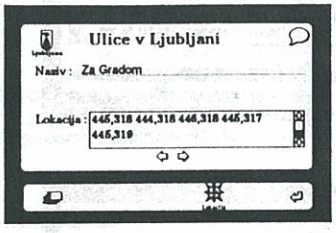

Ulice v Ljubljani
Naziv: Vozarski pot
Lokacija: 367,337 367,336 367,338 366,337
368,337 368,321 369,321 367,321

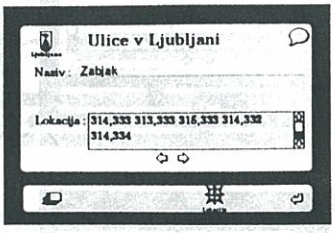

Ulice v Ljubljani
Naziv: Vrazov trg
Lokacija: 474,135 473,135 475,135 474,134
474,136

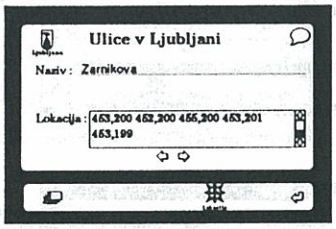

Ulice v Ljubljani
Naziv: Vrtaca
Lokacija: 58,235 59,235 57,235 58,234 58,236
81,235 81,236 81,234 80,235 82,235


Ulice v Ljubljani
Naziv: Vrtna ulica
Lokacija: 259,316 260,316 258,316 259,317
259,315 263,330 262,330 264,330

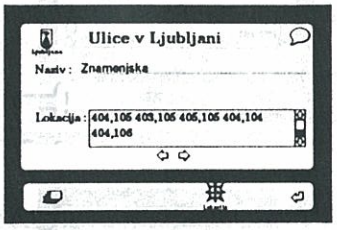

Ulice v Ljubljani
Naziv: Wolfova
Lokacija: 260,150 260,151 260,149 259,150 261,150

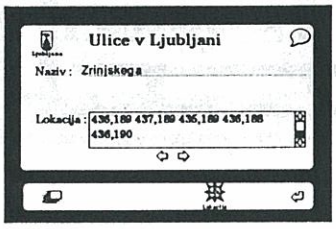

Ulice v Ljubljani
Naziv: Za Gradom
Lokacija: 445,318 444,318 446,318 445,317 445,319

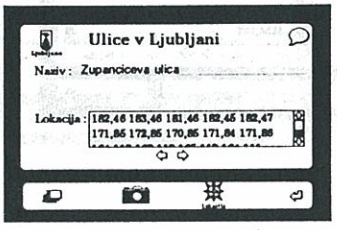

Ulice v Ljubljani
Naziv: Zabjak
Lokacija: 314,333 313,333 315,333 314,332 314,334

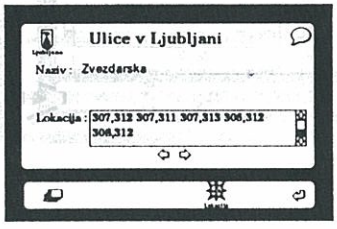

Ulice v Ljubljani
Naziv: Zarnikova
Lokacija: 453,200 452,200 455,200 453,201 453,199


Ulice v Ljubljani
Naziv: Zidovska
Lokacija: 258,221 258,221 257,221 258,222 258,220


Ulice v Ljubljani
Naziv: Znamenjska
Lokacija: 404,105 403,105 405,105 404,104 404,106


Ulice v Ljubljani
Naziv: Zrinjskega
Lokacija: 436,189 437,189 435,189 436,188 436,190


Ulice v Ljubljani
Naziv: Zupanciceva ulica
Lokacija: 182,46 183,46 181,46 182,45 182,47 171,86 172,86 170,86 171,84 171,86


Ulice v Ljubljani
Naziv: Zvezdarska
Lokacija: 307,312 307,311 307,313 308,312 308,312

Literatura

[1] A.L. Ambler, "Forms: Expanding the Visualness of Sheet Languages," *Proc. 1987 Workshop on Visual Languages,* Tryck–Center, Linkoping, Sweden, Aug. 1987, pp. 105–117.

[2] Apple Computer,Inc. *HyperCard User's Guide,* Cupertino, CA, Apple Computer, Inc.,1987.

[3] Apple Computer, Inc., *Inside Macintosh Volume V,* Addison–Wesly 1988, pp. V-13 – V-28

[4] E. Barrett, *Text, ConText, and HyperText : Writing with and for the Computer,* The MIT Press, 1985, pp. 77–92.

[5] A. Bornig, "Defining Constraints Graphically," *Proc. CHI 86,* Conf. Human Factors in Computing Systems, Apr. 1986, ACM, pp. 137–143.

[6] M.H. Brown and R. Sedgewick, "A System for Algorithm Animation," *Computer Graphics* (Proc. SIGGraph 84), Vol. 18, No.3, July 1986, pp. 177–186.

[7] "Scientific Visualization," *COMPUTER,* Vol.22, No. 8, Avgust 1989

[8] "Visualization in computing," *COMPUTER,* Vol.22, No. 10, October 1989

[9] F.P. Feynman, *Surely You're Joking, Mr. Feynman ! Adventures of a Curious Character,* Bantam Books, 1986, pp. 236–253.

[10] W. Finzer and L. Gould, "Programing by Rehearsal," *Byte,* Vol. 9, No. 6, June 1984, pp. 187–210.

[11] J.D. Foley and C.F. McMath, "Dynamic Process Visualization," *IEEE Computer Graphics and Applications,* Vol. 6, No. 2, Mar. 1986, pp. 16–25.

[12] E.P. Glinert and S.L. Tanimoto, "Pict: An Interactive Graphical Programming Environment," *Computer,* Vol. 17, No. 11, Nov. 1984, pp. 7–25.

[13] A. Goldberg, *Smalltalk–80: The Interactive Programming Environment,* AMC Trans. Programming Languages and Systems, Vol. 8, No. 4, Oct. 1986, pp. 419–490.

[14] D. Goodman, *'HyperCard' – The Complete HyperCard Handbook,* Bantam Books, 1987.

[15] A.N. Habermann and D. Notkin, "Gandalf: Software Development Environment," *IEEE Trans. Software Engineering,* Vol SE–12, No. 12, Dec. 1986 pp. 1117–1127.

[16] M. Hamilton and S. Zeldin, "Higher Order Software – Methodology for Defining Software," *IEEE Trans. Software Engineering,* Vol. SE–2, No. 1, 1976, pp. 9–32.

[17] J.C. Hart, D.J. Sandin, and L.H. Kauffman, "Ray Tracking Deterministic 3D Fractals," *Computer Graphics* (SIGGraph 89 Conf. Proc.) Vol. 23, No. 4, Aug 1989.

[18] E. Helttula, A. Hyrskykari, and K.–J. Räihä, "Graphical Specification of Algorithm Animations with Aladdin," *Proc. Hawai Int'l Conf. Systems Sciences Vol.II. Software Track,* CS Press, Los Alamitos, Calif., Order No. 1912, 1989, pp. 40–54.

[19] D.D. Hoffman, "The Interpretation of Visual Illusion," *Scientific American – Offprints,* Vol. 249, No. 6, Dec 1983.

[20] Y.–T. Hsia and A. Ambler, "Programming through Pictorial Transformations," *Proc. 1988 IEEE Int'l Conf. Computer Languages,* Oct. 1988, CS Press, Los Alamitos, Calif., Order No. FJ874, pp. 10–16.

[21] P.N. Johnson–Laird, *The Computer and the Mind: An Introduction to Cognitive Science,* Harvard University Press, 1988.

[22] Journal Harcourt, Brace, Jovanovich Publisher, *Journal of Visual Languages & Computing,* Academic Press

[23] S.M. Kosslyn, *Ghosts in the Mind's Machine: Creating and using images in the brain,* W.W. Norton &Co., 1983.

[24] B. Mihelič, *Vodnik po Ljubljani,* DZS 1989

[25] T.G. Moher, "Provide: A Process Visualization and Debugging Environment," *IEEE Trans. Software Eng.,* Vol. 14, No.6, June 1988, pp. 849–857.

[26] M. Moriconi and D.F. Hare, "The PegaSys System : Pictures as Formal Documentation of Large Programs," *ACM Trans. Programming Languages and Systems,* Vol. 8, No. 4, Oct. 1986, pp. 524–546.

[27] S.P. Reiss, "Pecan: Program Development Systems that Support Multiple Views," *IEEE Trans. Software Eng.,* Vol. SE–11, No. 3, Mar. 1985, pp. 276–285.

[28] S.P. Reiss, "Garden Tools: Support for Graphical Programming," in *Advanced Programming Environments,* Lecture Notes in Computer Science #244, R. Conradi, T. Didriksen, and D. Wanvik, eds., Springer–Verlag, N.Y., 1986, pp.59–72.

[29] G.–C. Roman and K.C.Cox, "Declarative Visualization in the Shared Dataspace Paradigm," *Proc. 11th Int'l Conf. Software Eng.,* CS Press, Los Alamitos, Calif., Order No. 1941, 1989, pp 34–43.

[30] R.V. Rubin, E.J. Golin, and S.P. Reiss, "ThingPad: A Graphical System for Programming by Demonstration," *IEEE Software,* Vol.2, No. 2, Mar. 1985, pp. 73–79.

[31] D. Shafer, *HyperTalk Programming,* Hyden Books, 1988.

[32] T. Teitelbaum and T. Reps, "The Cornell Program Synthesizer : A Syntax-Directed Programming Environment," *Comm. ACM,* Vol. 24, No. 9, Sept.1981, pp. 563–573.

[33] W. Teitelman and L. Masinter, "The Interlisp Programming Environment," *Computer,* Vol. 14, No. 4, Apr. 1981, pp. 25–33.

[34] TURBO PASCAL Ver. 5.5, *Object Oriented Programming Guide,* Borland International Inc. 1988.

[35] J. Voelcker, " 'PCs and workstations'—User power soars," *IEEE Spectrum,* Vol. 27, No. 2, February 1990, pp. 27.

[36] A. Wasserman and P. Pircher, "A Graphical, Extensible Integrated Environment for Software Development," *SIGPlan Notices,* Vol. 22, No. 1, Jan. 1987, pp. 131–142.

[37] R.B. Wilhelmson, "Numerical Simulations of Severe Storms," *Proc. Fourth Int'l Symp. : Science and Engineering on Cray Super Computers,* Cray Research, Oct. 1988.

[38] K.–H.A. Winkler and M.L. Norman, "Munacolor: Understanding High-Resolutions Through Color Graphics," *Astrophysical Radiation Hydrodynamics,* D.Reidel Publishing, 1986, pp. 223–243.

[39] K. Wright, "The Road to the Global Villege," *Scientific American,* Vol. 262, No. 3, March 1990, pp. 57–66.

[40] N.J. Zabusky, "Computational Synergetics," *Physics Today,* July 1984 reprint.

Zahvala

Zahvaljujem se mentorju gospodu Francu Solini za nesebično pomoč in mnoge koristne nasvete pri izdelave in oblikovanju moje diplomske naloge. Zahvaljujem se tudi gospodu Veselinu Miškoviču, NUK za pomoč pri zbiranju gradiva o Ljubljani in nasvete pri izdelavi informacijske aplikacije *Ljubljana*, ter gospodu Janezu Žibertu, Ada Graf za pomoč pri skaniranju slik, ki sem jih vključila v informacijsko aplikacijo *Ljubljana*.

Izjava

Izjavljam, da sem diplomsko delo izdelala pod vodstvom Franca Soline, docenta na Fakulteti za elektrotehniko in računalništvo Univerze v Ljubljani. Izkazano pomoč drugih sodelavcev sem v celoti navedla v zahvali.

Andreja Balon

Ljubljana, maj 1990